Three-dimensional modeling of human organs and its application to diagnosis and surgical planning

Bernhard Geiger

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Three-Dimensional Modeling of Human Organs and its Application to Diagnosis and Surgical Planning

Bernhard GEIGER

N° 2105
Novembre 1993

PROGRAMME 4

Robotique, image
et
vision

Three-dimensional modeling of human organs and its application to diagnosis and surgical planning

Construction et utilisation des modèles d'organes en vue de l'assistance au diagnostic et aux interventions chirurgicales

Bernhard Geiger
INRIA
BP 93
06902 Sophia Antipolis
France
email: geiger@sophia.inria.fr

Programme 4: Robotique, Image et Vision.

Résumé

Nous présentons une solution au problème de la reconstruction tridimensionnelle à partir de coupes parallèles. Après une définition d'une méthode exacte de liaison au plus proche voisin, nous montrons comment on peut calculer une approximation grâce à la triangulation de Delaunay. On présente des résultats obtenus à partir de données médicales.

La qualité des modèles tridimensionnels est ensuite évaluée sur des données synthétiques, et comparée avec une autre technique de reconstruction (*marching cube*).

Dans la troisième partie, on utilise la reconstruction d'un bassin et d'une tête fœtale pour simuler un accouchement. En faisant passer la tête par le bassin, on calcule la force résultante exercée sur la tête, et on définit une mesure de compatibilité fœto-pelvienne, qui pourra être intéressante pour décider de la nécessité d'une césarienne.

Mots-clés : reconstruction 3D, triangulation de Delaunay, diagramme de Voronoï, distance de Hausdorff, robotique, obstétrique.

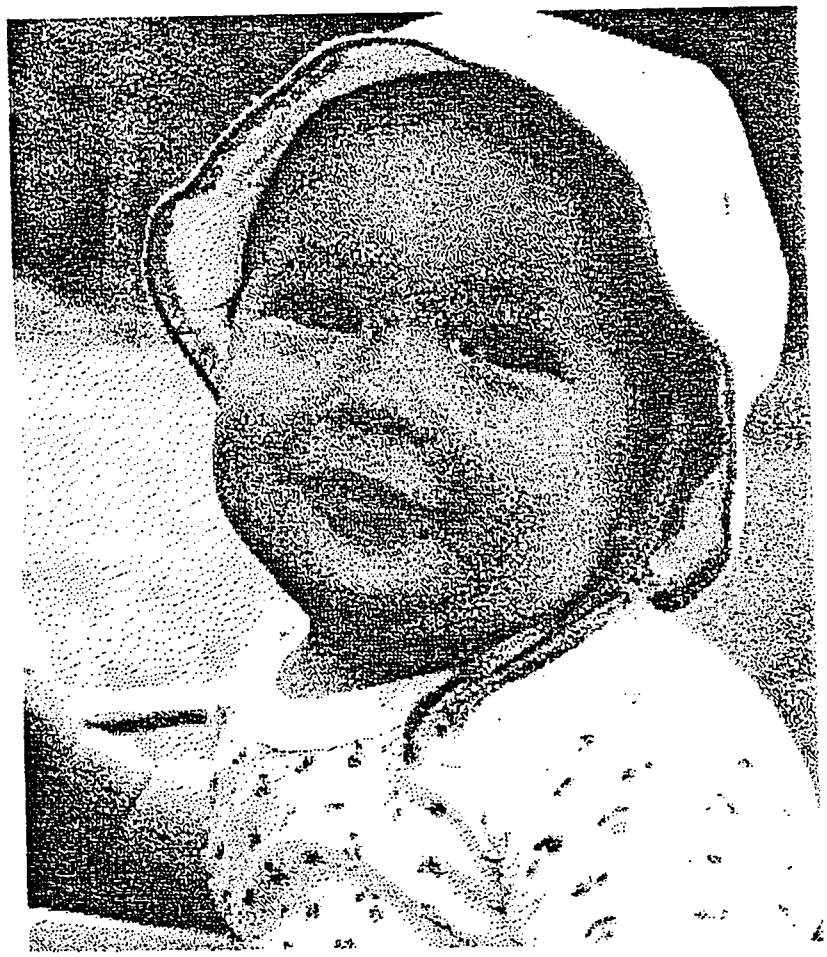
Abstract

We describe a method to obtain a 3D representation of objects given by a series of parallel cross-sections. We define a contour-to-contour connection based on geometric closeness and show a practical way to calculate an approximation using the Delaunay triangulation. The performance of our method is shown on several medical examples.

We further evaluate the precision of 3D models reconstructed from synthetic data and compare it to the *marching cube* reconstruction.

In the third part, we use the 3D model of a pelvis and a fetal head for simulating a delivery. We calculate the forces applying on the head and define a measure of adequateness. This measure may be interesting for the decision of making a cesarean section.

Keywords: 3D reconstruction, Delaunay triangulation, Voronoi diagram, Hausdorff distance, robotics, obstetrics.



Acknowledgement

I am deeply indebted to my supervisor Jean-Daniel Boissonnat. He gave me the opportunity to work at INRIA in a stimulating environment and on a fascinating topic.

I wish to express my gratitude to the committee chairman, Yves Rouchaleau, to Heinrich Müller, Steven Pizer and Claude Puech for reviewing my dissertation and to Nicholas Ayache, Philippe Cinquin, Heinrich Müller and Claude Puech for participating in the committee.

Thanks to Ron Kikinis from Harvard Medical School, Boston for providing the brain data, and especially to Frank Muennemann from Resonex Inc., Sunnyvale CA who sent me the pelvic data which was essential for the third part of my work. I would also like to thank Michael Stark from the Universität Dortmund for providing the marching cube data.

Nothing is impossible for XJPdraw, especially if the author of this drawing package, Jean-Pierre Merlet, is working in the next office and always ready to add new functionality.

I would like to thank Xuân-Nam Bui, Jacqueline Duquesne, Monique Teillaud and Mariette Yvinec for reading the drafts of this thesis and making many insightful comments and suggestions. Thanks to Nick May and David Hutber for proofreading the English version.

I would like to express my deepest appreciation to Herta and Philipp. Without their patience, this work would have been impossible.

Preface

This report is identical to my PhD thesis that was defended on April 6, 1993 at *Ecole des Mines de Paris.* The referees were Heinrich Müller Steven Pizer and Claude Puech, and the committee consisted of Yves Rouchaleau (chairman), Jean-Daniel Boissonnat (supervisor), Nicholas Ayache,Philippe Cinquin, Heinrich Müller and Claude Puech.

Contents

Introduction

Since the development of tomography, computers have been used extensively in medical diagnosis. Unlike classical radiography, tomography requires complex mathematical calculations in order to obtain a two-dimensional image. Computers are used for image treatment, visualization and archiving, and also for 3D reconstruction.

This technique, whereby a 3D model of certain organs is built from a number of 2D cross-sectional images, enables a better understanding of spatial structures, and also opens the way to new applications like automatic prosthesis milling, radiation therapy planning and surgical planning.

Methods for 3D reconstruction have existed for nearly 20 years. They may be classified into two groups: *surface oriented* methods and *volume oriented* methods. The reconstruction method that we present in the first chapter of this dissertation tries to combine the advantages of both groups. It is derived from the technique based on the Delaunay triangulation which was proposed by Jean-Daniel Boissonnat in 1984. We show its underlying theory and its application to tomographic data.

The second chapter concerns itself with the validation of 3D models. Surprisingly, this task has not received much attention in the past. Visualization is not sufficient for estimating quality; it may even be misleading, since many visualization algorithms like Gouraud shading are designed to extrapolate properties. If 3D models are used for purposes other than pure visualization, the precision may be crucial. Maybe this work could contribute to the development of benchmark tests in the future, which would allow the comparison of the different reconstruction methods.

Simulation of delivery is a task that touches two domains: computational geometry for 3D modeling, and robotics for motion planning. We show in the third chapter that the results of the Delaunay reconstructions can be easily adapted for this application.

Chapter 1

The Delaunay Reconstruction

1.1 Introduction

An important problem in medical imaging is the creation of three dimensional models from a set of tomographic cross-sections. Such reconstructions of human organs are used for surgery planning, prosthesis milling, radiation therapy planning and volumetric measurements. Because of the keen interest, there have been a great number of approaches, which may be roughly classified into two groups: *surface reconstruction* and *volume reconstruction.*

Historically, surface reconstruction was the first approach. In surface reconstruction, the problem consists in building a surface between contours in adjacent cross-sections. This is usually done by connecting contour segments to vertices in the opposite plane. In this way, the surface is defined by a set of triangular tiles. Keppel [18] was the first to reduce the problem of connecting vertices to a search in a toroidal graph. He used a maximum volume heuristic to select a path. Fuchs, Kedem and Uselton [13] presented the first efficient algorithm, also based on a search in a toroidal graph. Their method generalizes the results of [18] and can be used with various optimization criteria. They proposed a minimal surface heuristic, but other optimization criteria may be used.

These solutions were limited to one single contour on each cross-section. Several authors proposed methods to solve the branching problem in the case of multiple contours in each plane. A method that handles simple branching cases was described by Christiansen and Sederberg [10]. In that approach, complex contours require user interaction. Other solutions came from [27, 28, 29, 20].

Even if restricted to one single contour in each cross-section, the problem may have no solution, or—if it exists—it may not be unique. Gitlin, O'Rourke and Subramanian [24] have recently shown an example of a pair of contours that cannot be connected to a simple polyhedron. We present in Appendix A the proof that the problem can always be solved by adding at most two vertices onto

the contours.

The advantages of surface reconstruction are as follows:

- Usually, the methods are very fast.
- These methods reduce the amount of data by selecting two-dimensional representations of object surfaces.
- A 2D representation—especially a polyhedral representation with triangular faces—allows fast display methods, Gouraud or Phong shading and high-quality rendering, for example with ray-tracing methods.
- The methods may be adapted to particular objects by choosing one of several possible heuristics.
- Since contours are linearly interpolated, there is no need for anti-alias steps.
- Contours may be processed (spline interpolation, vertex reduction).

The disadvantages of surface reconstruction are as follows:

- Several methods are limited to a single contour in each cross-section. Only simple branching is possible.
- In realistic scenes, there is a large variety of topologies (multiple branching, multiple holes). A method that is constrained to contour lines cannot adapt to all possible cases.
- In addition to the data segmentation, we have to add a contour tracking p rocedure with consistency check, since most methods are limited to simple closed contours.

The voxel-technique is a volume based approach. A voxel is the spatial equivalent to a pixel. Since scanner images are arranged on a regular 2D grid (the pixels) it is quite natural to extend them to volume elements (Barillot et al. [2]). There are two ways to display such a set of parallelepipeds: One can fit a surface on it (Marching Cube [23]) and render the object with conventional surface-rendering algorithms, or surface normals can be deduced directly from the voxel data (Höhne et al. [17], Levoy [22]). The second approach is usually called volume rendering. The advantages of the voxel methods are as follows:

- The simple heuristic based on geometric closeness.
- Volume rendering omits the need for additional processing, for example to track a surface and fit geometric primitives on it.

The disadvantages of the voxel methods are as follows:

- The large volume of data that has to be manipulated.
- No simple defined surface is available for rendering.
- If cross-section distances are large compared to pixel distances, an interpolation step is necessary to avoid jagged shapes.

Another volume approach is the Delaunay reconstruction proposed in [4]. In that method, the volume elements are not equally shaped, but consist of tetrahedra that are adapted to the object shape. In this way, some of the advantages of surface reconstruction—the important data reduction and the polyhedral representation—are combined with the strength of the voxel-method: the handling of complicated topologies.

In this chapter, we define the Delaunay reconstruction which is based on Boissonnat's approach [4]. We consider the reconstruction from a point of view that allows us to compare its heuristic to that of the voxel technique.

We give a brief description of the reconstruction problem in the following section. In Section 1.3, we recall the basic definition of Voronoi diagrams and the Delaunay triangulation, and show the principal underlying mechanisms of to obtain a connection based on geometric closeness. A detailed description of our method is given in Section 1.4. Experimental results and a concluding section are presented at the end of this chapter.

1.2 The reconstruction problem

Tomographic data usually consists of a series of cross-sectional images, where each pixel represents a characteristic value inside the body. This value may be protein density in MRI, or X-ray absorption in CT images. The typical image size is 256×256 or 512×512 pixels. The image to image distances or cross-section distances are often significantly greater than the 2D image resolution. The reasons are physical constraints, time constraints or to avoid excessive irradiation with X-ray techniques. Recently developed MR scanners, however, are able to reach a slice distance equal to the in-plane resolution. For conventional diagnosis, the pixel values are mapped to gray values or to appropriate colors, and displayed on a screen or fixed on a film.

The procedure of 3D reconstruction can be divided into 4 steps (Figure 1.1):

1. Image segmentation
2. Region labeling

3. Global connection

4. Local connection

The first two steps can be considered as preprocessing steps, using classical image processing techniques.

1. Image segmentation
In this step, the pixels representing one tissue type have to be isolated. There is a large variety of image enhancement and segmentation techniques, depending on tissue type and imaging modality. Bone in CT images and brain tissue in MRI are examples of highly contrasted data that can be automatically segmented. Segmentation of other objects, however, may require human assistance.

2. Region labeling
Segmented regions of the same tissue type are further classified into anatomical structures. We may for example label the left and the right ventricle of the heart, or arteries and veins. This additional information is not included in the scanner data. Consequently, this procedure needs human interaction, or knowledge based methods together with shape recognition.

3. Global connection
In this step, we decide which regions will be connected from image to image. Each segmented and labeled 2D region will be connected to zero, one or several 2D regions in the adjacent cross-sections. Like the previous step, this procedure requires a priori information. This step is also called *the correspondence problem* in [28].

4. Local connection
Finally, for each region to connect, we must decide how to link the region parts in detail. The problem consists in finding a way to split the regions, especially when there are multiple branchings. For surface tiling methods, we have to choose triplets of vertices to build the surface triangles. This task is usually too complex to be driven interactively.

The different reconstruction methods now cover one or several of these steps. The requirements of a particular method depend on additional information like:

- Cross-section distance
- Object size
- Tissue type
- Purpose of reconstruction
- Other a priori knowledge

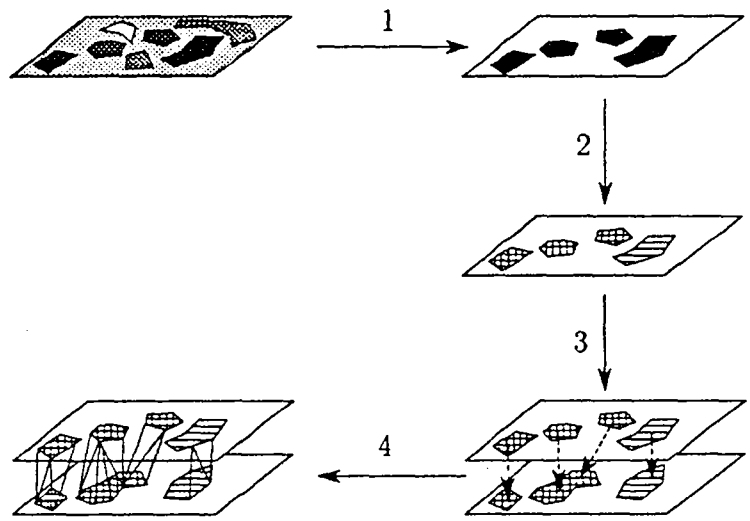


Figure 1.1: 1) Segmentation 2) Labeling 3) Global connection 4) Local connection.

A reconstruction method based on geometric closeness—like the voxel method that connects overlapping regions, or our own method—can perform steps 2 to 4 on simple objects. If there is only one simple region in each cross-section, or if several regions are sufficiently far apart, region labeling may be omitted, and global and local connections are determined by closeness (see Figure 1.2 (a) and (b)). However, the regions in Figures 1.2 (c) and (e) may yield wrong connections. This example could represent arteries and veins, which usually lie close to one another. If the purpose of the reconstruction is the visualization of spatial relationships, such short-circuits may be acceptable; if the models are used for other applications like volume calculation or blood flow simulation, correct connectivity is necessary. In this case, the contour labeling cannot be omitted. After a separation of the different labeled regions and two separate reconstructions, the result of a closeness (or "nearest neighbor") method would be correct.

For objects, where the required connections are not identical to simple heuristics like closeness, we have to apply special reconstruction methods. This occurs with contours, where special features have to be connected correctly (see Figure 1.2 (d) and (f)). As an example, consider complex brain contours. This case can only be solved by special techniques, which, amongst other things, may detect similar curvature.

We suppose in the sequel that

- Image segmentation and region labeling have been applied.
- The input regions have been restricted to one tissue type of one label if

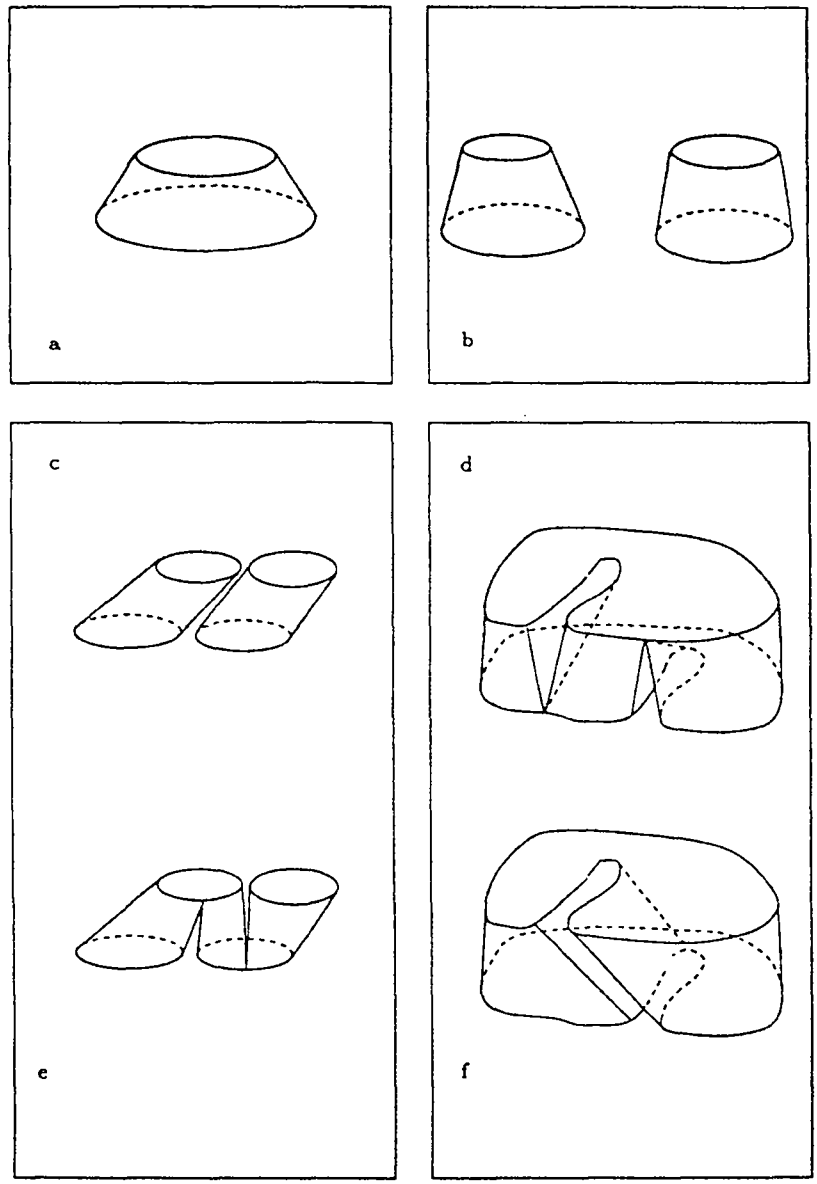


Figure 1.2: Several examples of connecting contours on two adjacent cross-sections.

necessary (for example, arteries have been separated from veins, if correct connectivity is required).

- We have no additional a priori knowledge about the necessity of constrained connections of contour details.

A 3D object is intersected by a number of parallel planes. Since concave objects are allowed, we may get one or several regions in each plane. The contours of those regions may be approximated to by polygons, possibly some of them lying inside others. We now want to link these regions (or to connect the contours) in such a way as to obtain the "best" approximation to our original object. The difficulty consists in finding criteria that will produce a "good" approximation. All our information is concentrated in the cutting planes; we know the object shape in each cross-section. Consequently, one condition the reconstructed object has to satisfy is that if we cut it along the original planes, we get the actual regions. This condition however defines no unique solution. Figure 1.2 and Figure 1.3 show possible solutions on identical contour sets that satisfy it. Selecting a particular solution is a heuristic decision, if no a priori information is present.

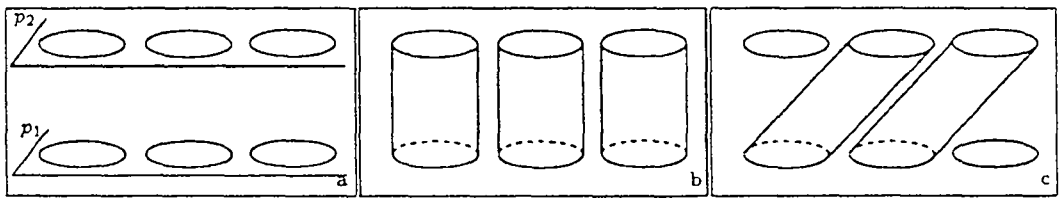


Figure 1.3: A set of 6 contours, pairwise superimposed in two horizontal, parallel planes p_1 and p_2. (b) and (c) show two possible connections.

All reconstruction methods, except the interactive ones, make such heuristic assumptions. Typical heuristics are: maximizing volume, minimizing surface, minimizing edge length and minimizing angles. In voxel reconstruction, the heuristic consists of extending all object pixels to a parallelepiped of height equal to the slice distance, thus favoring "vertical" connections, especially connections of overlapping parts. It is quite a simple and natural assumption, and with an adequate relationship between object size and slice distance, the results are very satisfactory.

In our approach, we subdivide the planar regions into triangles, and extend these triangles to tetrahedra by linking them to the contours in adjacent cross-sections. We attempt to achieve an almost vertical, "voxel-like" connection, but at the same time to reduce the data, avoid anti-alias or interpolation steps, and get directly to a polyhedral representation. The data reduction effect is evident, since

a triangle in a cross-section covers an area of several pixels, and a tetrahedron in 3D occupies the space of several voxels. The other important goal—the vertical connection— can be accomplished with the help of Voronoi diagrams, as we will see in the following section.

1.3 Basic concepts for closeness

1.3.1 Voronoi diagram and Delaunay triangulation

In this subsection, we recall the definitions of Voronoi diagram and Delaunay triangulation, and show the relationship to the medial axis. A survey on Voronoi diagrams can be found in [1].

Given a set S of N sites $S = \{p_i \in E^2 | i = (1..N)\}$ such that no four sites lie on a common circle.

We define $V(i)$ as the set of points closer to site p_i than to any other site in S. $V(i)$ is called the *Voronoi cell* associated to p_i (See Figure 1.4). The union $V(S)$ over all $V(i)$ is called the *Voronoi diagram* of S. It can be considered as a subdivision of the plane in convex polygonal regions—the Voronoi cells. The boundaries of the regions are referred to as *Voronoi edges*, the joints of (always) three Voronoi edges are called *Voronoi vertices*. Each Voronoi edge is associated with two adjacent Voronoi cells, and is a part of the perpendicular bisector of their sites, thus denoting the locus of equal distance. A Voronoi vertex is equidistant to three sites.

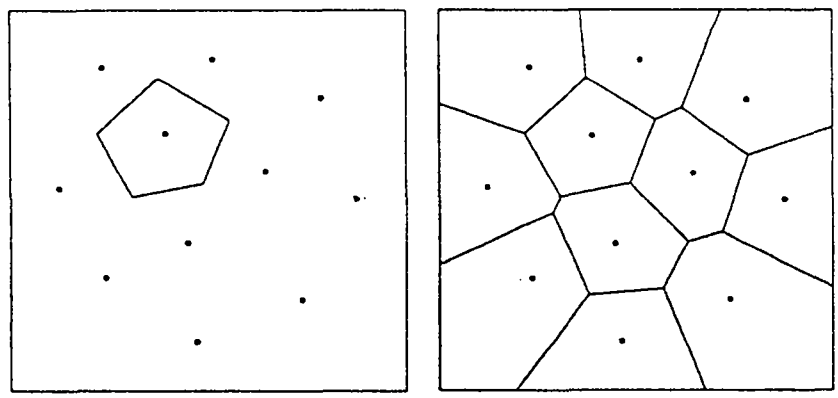


Figure 1.4: A Voronoi cell (left) and the Voronoi diagram (right) of a number of sites.

If we draw a line segment between each pair of sites whose Voronoi cells share an edge, we obtain a triangulation of the points in S called the *Delaunay*

triangulation (See Figure 1.5). A Voronoi vertex represents a Delaunay triangle; more precisely, it is the center of its circumcircle. Each Voronoi edge corresponds to an edge in the Delaunay triangulation despite the fact that they may not even intersect. This geometric difference between Voronoi diagram and Delaunay triangulation becomes important in the reconstruction issue. Some further properties of the Delaunay triangulation are:

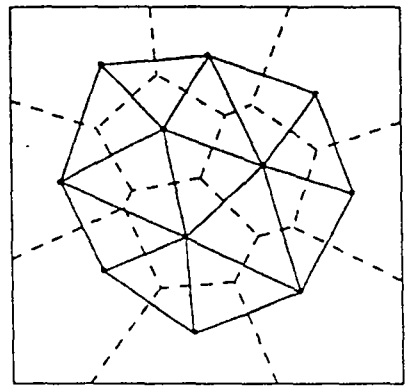


Figure 1.5: A Voronoi diagram (dashed) and its straight-line-dual, the Delaunay triangulation.

- The boundary of the Delaunay triangulation is the convex hull of its sites.
- The Delaunay triangulation maximizes the minimum angle over all triangulations. Therefore, the shape of Delaunay triangles is in general compact.
- The number of triangles in the Delaunay triangulation is at most $2N - 5$, where N is the number of vertices in the triangulation.
- A Delaunay triangle does not contain any other site in its circumcircle (empty circle property). This implies that any triangulation without obtuse angles must be Delaunay.
- There exist several well known algorithms to calculate it.
- Voronoi diagrams can be used to calculate fast point location.

The definition of Voronoi diagrams and Delaunay triangulation in 3D space is straightforward (See for example [25]). Instead of triangles we will get tetrahedra, besides Voronoi edges there will also be Voronoi faces, and the empty *circle* property translates in an empty *sphere* property.

Several generalizations of the Voronoi diagram may be defined. One generalization is the *edge Voronoi diagram* (EVD), where the defining sites are not points but straight line segments (see [19], [30], [12]). In this case, the Voronoi diagram is built of straight and parabolic edges. The edge Voronoi diagram of a polygon is also called *medial axis*. The medial axis of a polygon is the locus of points equidistant to at least two contour points, and closer to that contour points than to any other. This is a well studied construct in pattern recognition, since it closely represents the metric and topological properties of shapes (cf. [3], [21]). Another generalization is the *region Voronoi diagram* (RVD) where the sites are simple closed polygonal regions. The result is similar to the edge Voronoi diagram—straight or parabolic edges—with the difference that the inside of the polygons would be filled (Figure 1.6). Correspondingly, one can define an *edge Delaunay triangulation* or *region Delaunay triangulation*, which would be composed of triangles and trapezoids.

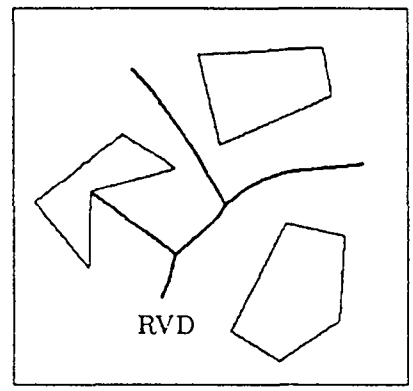


Figure 1.6: A region Voronoi diagram of three polygonal contours.

Similar to the Voronoi diagram of point sites, the region Voronoi diagram and the edge Voronoi diagram can be extended to 3D. Such a construct would be composed of parabolic surfaces.

1.3.2 Contour closeness

The reason for applying a Voronoi diagram is its underlying notion of geometric closeness. Similar to the voxel-heuristic, we would like to achieve a connection between overlapping contour regions. In addition, the non-overlapping parts have to be connected in an intelligent way in order to avoid jagged shapes. The general

idea is to subdivide the regions in appropriate parts, and link them to the nearest neighbor regions in the adjacent planes. If we define for example a connection that links each infinitesimal region part δr_1 to its nearest region part δr_2 in the adjacent plane, we would get a result as shown in Figure 1.7. Overlapping parts are connected vertically, non-overlapping parts are connected to their nearest neighbors.

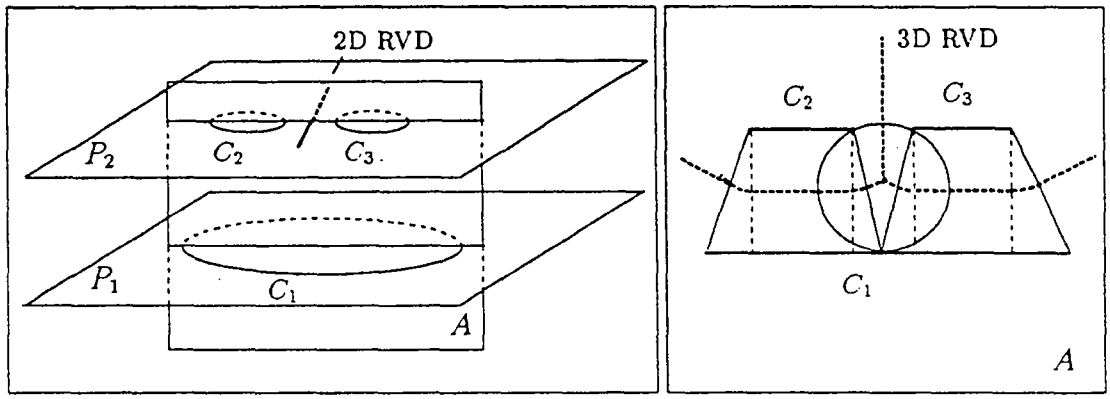


Figure 1.7: Three contours in two adjacent planes: C_1 in P_1, C_2 and C_3 in P_2. The righthand drawing shows the intersection with the vertical plane A. The intersection with the 3D region Voronoi diagram is dotted bold.

Such a solution can be obtained by understanding each contour polygon as site, and calculating the region Voronoi diagram RVD in each cross-section. The orthogonal projection of each RVD onto the adjacent plane determines then the subdivision and the connections of the polygons. Figure 1.8 illustrates this behavior: the RVD separates P_2 into the set of points closer to C_2 and the set of points closer to C_3. An infinitesimal surface δr of C_1 would be connected to C_2 if it lies on the left side of the orthogonal projection of the RVD, otherwise to C_3. The contour C_1 hence is divided along the projection of the RVD of the adjacent cross-section. Since this projection may consist of parabolic edges, a connecting surface between adjacent cross-sections would contain parabolic patches. But also a connection between non-parallel straight lines—one in each cross-section—would result in a hyperbolic paraboloid (see Figure 1.9).

So, besides the complexity, this method has a major inconvenience: the reconstructed surface would not be polyhedral, but built of parabolic faces. We therefore chose a different method. We show how to calculate an approximation of the above outlined method, by calculating the Voronoi diagram of point sites. The advantages are:

- We obtain a simple subdivision of the contour regions into triangles.

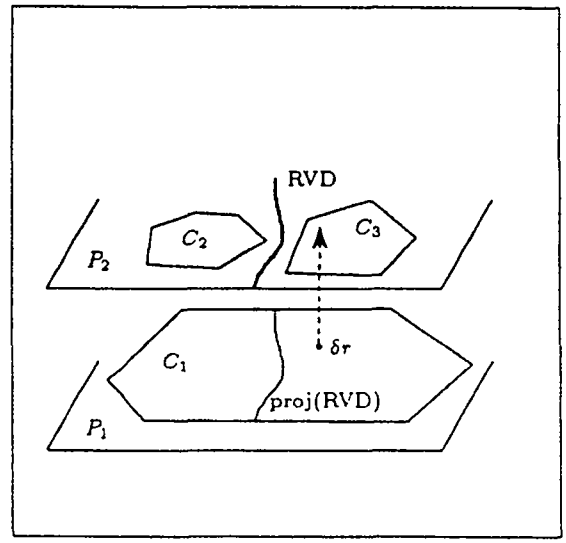


Figure 1.8: Connecting contours between adjacent cross-sections by calculating the region Voronoi diagram (RVD) and linking infinitesimal surfaces to the closest point.

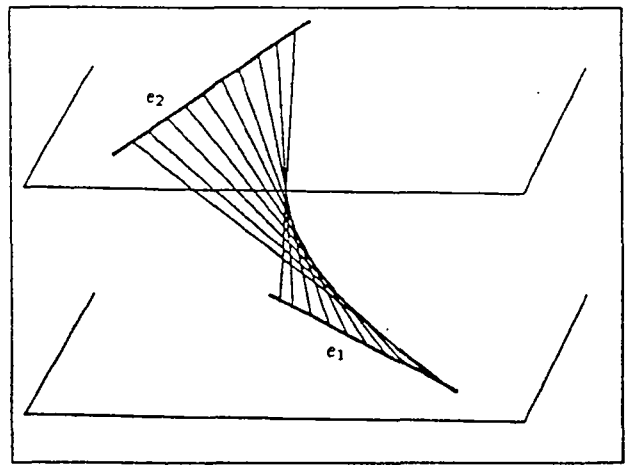


Figure 1.9: Connecting each point of e_1 with the closest point of e_2 yields a hyperbolic paraboloid.

- The 3D elements are tetrahedra.
- There is no need for parabolic edges and faces.
- The final result will be a polyhedron.

There is another reason for calculating rather an approximation than an exact solution: the input already consists of polygonal approximations of the real object contours. So the effort of doing an exact calculation based on already approximated data would be inadequate.

1.3.3 Approximation by a regular mesh

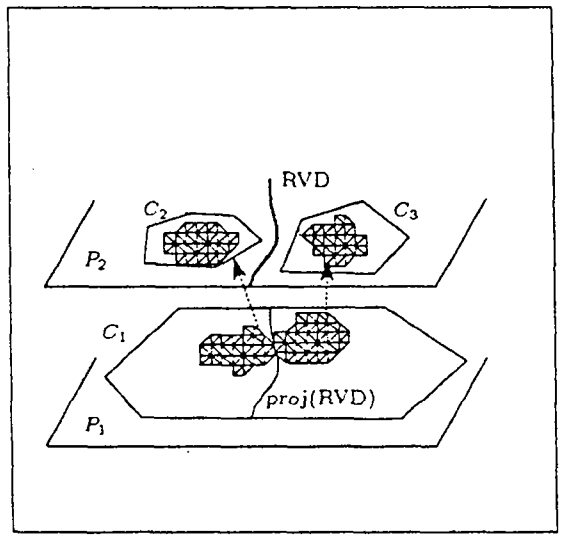


Figure 1.10: Contours are filled with a triangular mesh.

The first simplification that we propose is to enlarge the infinitesimal contour parts to small, non-obtuse triangles. Therefore, we have to add vertices onto and inside the contours. The construction of such a mesh is often required in finite element calculation. A solution by successive refinement is shown in [16]. Because such a mesh satisfies the empty circle condition, it is a Delaunay triangulation.

Instead of connecting each infinitesimal contour part to the closest infinitesimal contour part in the adjacent section, we now connect each triangle to a vertex in the opposite cross-section. The vertex is selected according to a closeness criterion—the closeness to the circumcenter of the triangle. Each triangle

gets connected to the vertex closest to its circumcenter [1]. As shown in Figure 1.10, the contour division is similar to the first approach. The triangles in P_1 are connected to one of the two contours in P_2, depending on the position of their circumcenters. However, the separation line is now a polygonal approximation of the RVD. The quality of the approximation depends on the size of the triangles. The advantage so far is to avoid parabolic surfaces. Still this method is not satisfying because of the quadratic complexity to construct such a regular mesh. Fortunately, filling regions with a regular mesh is not the only way to obtain a nearest neighbor connection. Any triangulation results in a similar connection

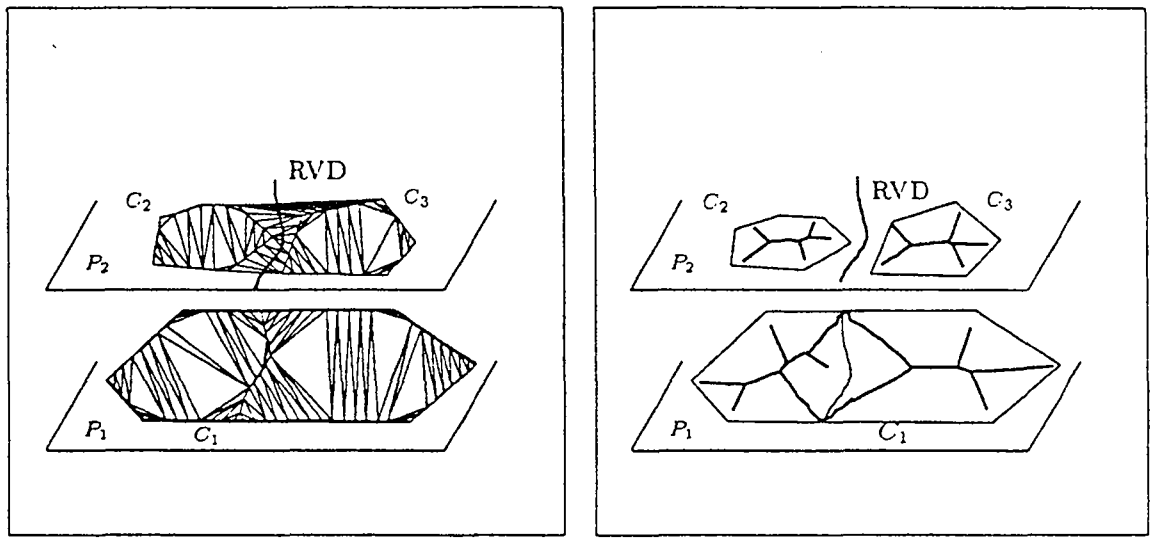


Figure 1.11: The Delaunay triangulation (left) and the Voronoi diagram of contour vertices (right). Vertices have been added to the contours segments and onto the projection of the adjacent RVD.

if it satisfies the following conditions:

1. The triangulation enables a separation along the projection of the adjacent RVD.

2. The circumcenters of the triangles stay inside the contours or inside the contour parts divided by the projection of the adjacent RVD.

[1] Connecting the triangles in the cross-sections to vertices in the adjacent cross-sections is not sufficient to obtain a compact tetrahedrization, as we will see in Section 1.4

The second condition is necessary because of our closeness criterion which is related to the circumcenter. Figure 1.11 shows such a triangulation. It has been obtained by adding equally spaced vertices onto the contours and onto the projection of adjacent RVD and calculating the Delaunay triangulation. Condition 1 can be satisfied by adding vertices onto the projection of adjacent Voronoi diagrams. Condition 2 can be satisfied by adding vertices onto the contour segments.

The general idea of our reconstruction method is now: we start with a 2D Delaunay triangulation of the contour vertices. We add vertices onto and inside contours and update the triangulation until the above conditions are satisfied. Then we pass from 2D to 3D by linking the triangles in each cross-section to tetrahedra. Changing from the RVD to the simpler Voronoi diagram of point sites however requires additional steps, as we will see in the following subsections.

1.3.4 Contour containment

In our application, the object contours are given as a set of straight line segments, forming one or several simple closed polygons. However, just triangulating the polygons vertices—or calculating the Voronoi diagram of point sites—may result in a triangulation where contour segments are not guaranteed to be edges of the triangulation. Since our goal is to get a 3D polyhedron whose intersection with the given cross-sections yields the original contours, our triangulation has to satisfy the following requirement:

All contour segments have to appear as Delaunay edges in the Delaunay triangulation (contour containment condition).

To obtain a triangulation that satisfies this condition, we simply calculate the Delaunay triangulation of the polygons' vertices. Then we check each contour segment to be contained in the Delaunay triangulation. Segments that do not appear as Delaunay edges are split into two parts by adding a new vertex in the middle. We add the new vertices to the Delaunay triangulation and verify the contour containment condition once more. It can be shown that such a procedure terminates and yields a Delaunay triangulation that satisfies the containment condition ([4]). The contour shape is not changed since we add vertices onto contour segments (see Figure 1.12).

1.3.5 Internal and external Voronoi skeleton

Once the containment condition is satisfied, we can divide the Delaunay triangles in two groups: *internal triangles* lying inside the contours and *external triangles* lying outside the contours. We call *internal Voronoi skeleton* the set of edges of

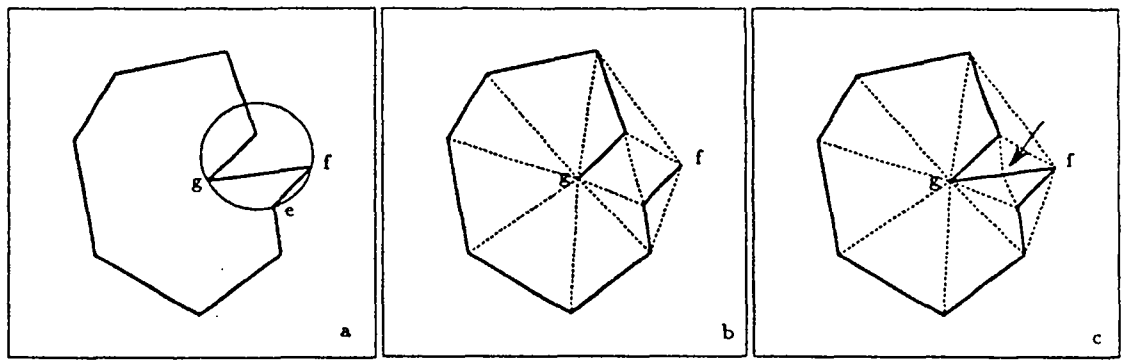


Figure 1.12: (a) A contour. It is evident, that triangle e,f,g cannot be part of a Delaunay triangulation, since its circumcircle would contain another site. (b) shows the Delaunay triangulation of the contour points. Delaunay edges which are incident with contour segments are drawn bold. Contour edge f,g is contained after addition of a vertex to f,g (c).

$V(S)$ which are dual to an internal Delaunay edge (that is an edge shared by two adjacent internal Delaunay triangles). Similarly, we define *external Voronoi skeleton* the set of edges of $V(S)$ which are dual to an edge shared by two external Delaunay triangles (see Figure 1.13). We call *internal Voronoi vertices* resp. *external Voronoi vertices* the Voronoi vertices dual to internal resp. external Delaunay triangles. The Voronoi edges that neither belong to the external Voronoi skeleton nor to the internal Voronoi skeleton are dual to contour edges. We call them *interposed Voronoi edges*, because they either connect the internal Voronoi skeleton to the external Voronoi skeleton or are infinite Voronoi edges. The internal Voronoi skeletons represent the contour polygons, but as we mentioned above, the Delaunay edges and the Voronoi edges of a point set may be quite distant. The internal Voronoi skeleton is not guaranteed to lie inside the polygon boundaries. Even worse, polygons with the same shape but having a different distribution of vertices may have totally different Voronoi skeletons (see Figure 1.15). A Delaunay triangulation of contours whose internal Voronoi skeletons lie partly outside the contours would violate the condition 2, because the circumcenters do not stay inside the contour regions. To obtain a closest contour connection, such a triangulation must be further improved.

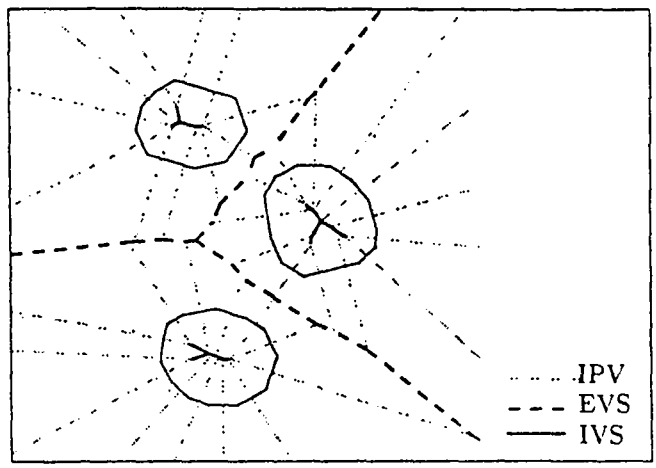


Figure 1.13: Internal (IVS) and external (EVS) Voronoi skeleton and interposed Voronoi edges (IPV) of three contours.

1.3.6 Eliminating obtuse triangles

In general, increasing the number of polygon vertices makes the Voronoi skeleton tend towards the medial axis of the polygon (see Figure 1.14). The convergence of such a process is shown in [26]. Suppose we add n vertices, equally spaced, onto the contour. If we increase n, we get an approximation of the medial axis. If several contours are present in one cross-section, the external Voronoi skeleton or external medial axis forms "macro" cells, enclosing the contours. The points inside such a "macro" Voronoi cell are closer to its contour than to any other contours. Hence the external Voronoi diagram approximates to the locus of equidistance from at least two contours, the RVD (see Figure 1.13).

Adding a number of equally spaced vertices to the contours therefore would be a possible solution to the problem of obtaining a nearest neighbor reconstruction (condition 2). We propose another solution that increases the quality of shape representation of the Voronoi skeletons by adding a smaller number of vertices. To guarantee that the internal Voronoi skeleton lies inside its contour polygon, (and also that the external Voronoi skeleton lies outside the contours), we add new vertices to contour segments belonging to obtuse Delaunay triangles. For each contour segment, we consider the opposite angles of the two triangles it belongs to. If one is greater than 90 degrees, its corresponding Voronoi vertex lies outside the triangle. In this case we add a new point at the normal projection of the opposite vertex onto the contour segment. This will divide the obtuse triangles into two right-angled triangles (see Figure 1.15). Note that this procedure does not eliminate all obtuse angles in the Delaunay triangulation, as only

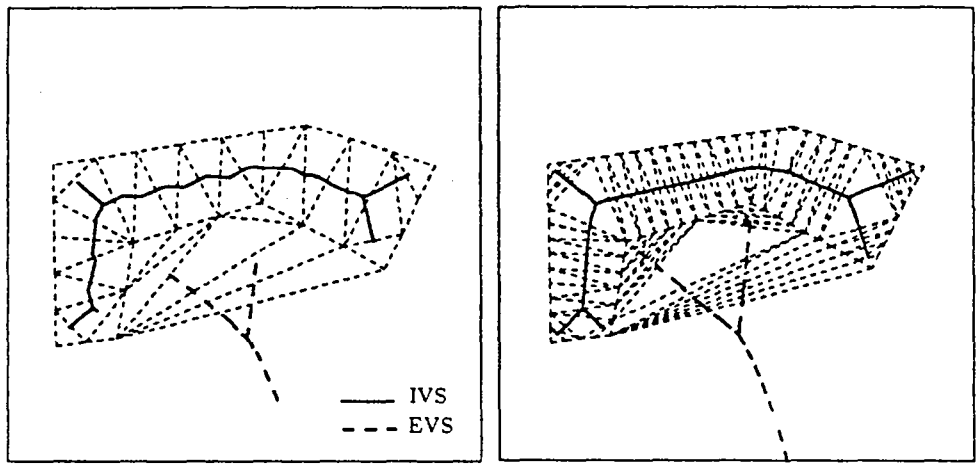


Figure 1.14: Two approximations of the medial axis, with 27 (left) and 72 (right) vertices on the contours.

obtuse angles opposed to contour segments are detected. But this is sufficient to guarantee that the internal Voronoi skeletons stay inside its contours.

1.3.7 Adding adjacent Voronoi skeletons

We have shown in Section 1.3.3 why it is necessary to add vertices inside the contours. We have to project the region Voronoi diagram (or external medial axis) of the adjacent cross-sections onto the contours and add vertices to this line. A subsequent Delaunay triangulation will contain edges connecting these vertices, and hence enable a separation along an approximation to the external medial axis (condition 1 for nearest neighbor connections). Consequently, instead of calculating the region Voronoi diagram (or external medial axis), we may directly use the external Voronoi skeleton that results from the Delaunay triangulation of the contour vertices, since we have shown above that the external Voronoi skeleton is an approximation of the medial axis.

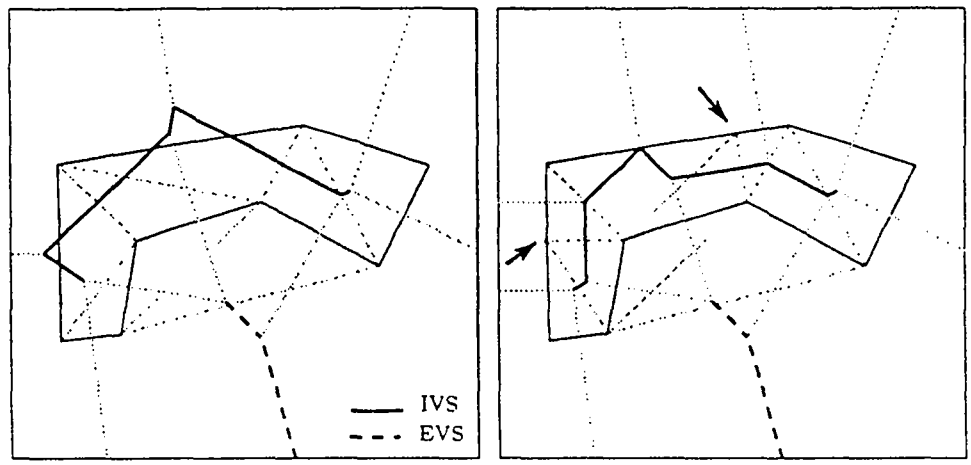


Figure 1.15: Two vertices have been added to the contour in the righthand image to avoid obtuse angles.

1.4 The Delaunay reconstruction

We reduce the problem of finding an overall reconstruction to computing a solid slice between each pair of adjacent cross-sections. In each cross-section, our input consists of one or more closed simple polygons, which may possibly be nested. The contours are oriented in such a way that the interior of the object they are describing is on its righthand side, the exterior on its lefthand side.

Our algorithm consists of three major steps: in the first step, we calculate a 2D triangulation of the contours, in the second step, we map the triangles to tetrahedra. In the third step, we have to delete the tetrahedra which cannot belong to the reconstructed object.

1.4.1 First step: 2D triangulation

Unlike to the voxel-method, where the cross-sections are subdivided into regular elements (pixels), we divide the cross-sections into triangles by calculating the Delaunay triangulation of the contour vertices. If necessary, additional vertices are added on the contours, to satisfy the contour containment condition and to avoid obtuse angles, as described in the previous sections (see Figure 1.16). We then add vertices onto the orthogonal projection of the external Voronoi skeletons in adjacent cross-sections and update the triangulation.

The first advantage is a considerable data reduction, since the number of

triangles is linearly related to the number of vertices. To represent the object regions by a regular grid would cost space in quadratic terms.

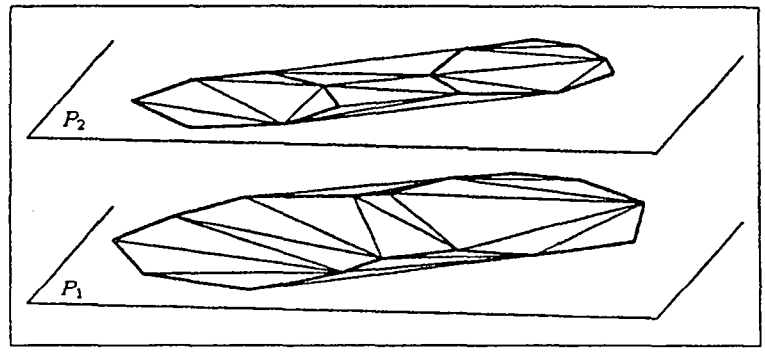


Figure 1.16: Delaunay triangulation of contours in two adjacent planes.

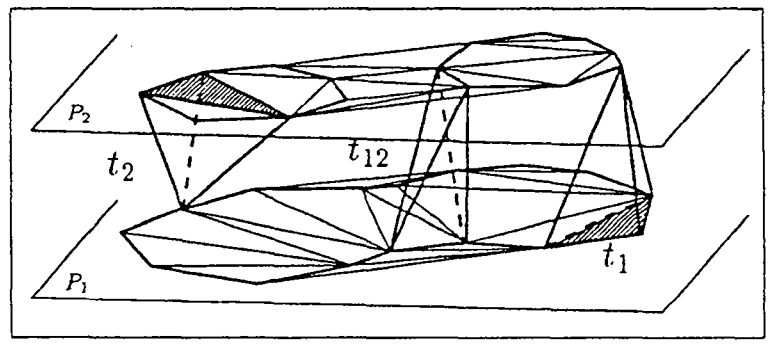


Figure 1.17: Connecting triangles to tetrahedra.

1.4.2 Second step: 2D to 3D mapping

In this step, we extend the triangles in two adjacent cross-sections to tetrahedra connecting the sections. If we consider two adjacent cross-sections, say P_1 and P_2 then for each triangle $t \in P_1$ we search the vertex $v \in P_2$ which lies closest to the circumcircle of t. We connect t with v to obtain a tetrahedron. Similarly, each triangle $t \in P_2$ is connected to vertex $v \in P_1$ which lies closest to the circumcircle of t.

Up to now our 3D triangulation consists of tetrahedra which we call of type t_1 or t_2, depending on whether they have a face in P_1 and a vertex in P_2, or a

face in P_2 and a vertex in P_1. The third kind of tetrahedron, called t_{12}, having one edge in P_1 and one edge in P_2, can be found by calculating the intersections of the Voronoi diagram of P_2 projected orthogonally onto the Voronoi diagram of P_1. We define the graph $\mathcal{G}$ as the union of two Voronoi diagrams $V_1 \in P_1$ and $V_2 \in P_2$, projected orthogonally onto the same plane. All topological and geometric information is then contained in that graph. The vertices of $\mathcal{G}$ are:

- The vertices of V_1 representing t_1 tetrahedra.
- The vertices of V_2 representing t_2 tetrahedra.
- We add a vertex on each intersection of an edge of V_1 with an edge of V_2. These vertices represent t_{12} tetrahedra. Such a t_{12} tetrahedron is constructed by linking the two Delaunay edges dual to the intersecting Voronoi edges.

We call the different vertices of $\mathcal{G}$ in the sequel t_1, t_2 or t_{12} nodes. If two nodes of $\mathcal{G}$ are connected by an edge, the represented tetrahedra share a face. We call an edge of $\mathcal{G}$ *belonging to a Voronoi skeleton* if it is a part of a projection of a Voronoi skeleton. We may further separate the edges of $\mathcal{G}$ into belonging to internal or external Voronoi skeletons of P_1 and P_2. Figure 1.18 shows an example of a graph $\mathcal{G}$ and its corresponding tetrahedrization.

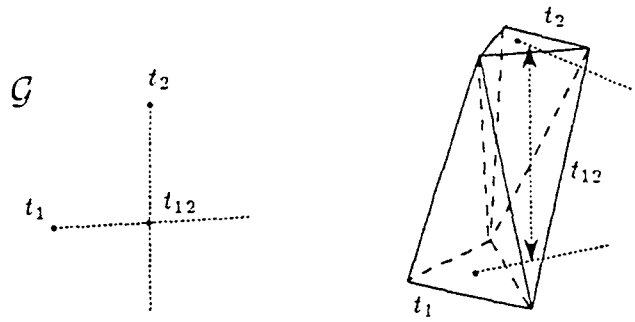


Figure 1.18: An example of a graph $\mathcal{G}$ (left) and its corresponding tetrahedrization (right). The arrows indicate the intersection of the orthogonal projection.

As shown in [4], the resulting triangulation is in fact the 3D Delaunay triangulation of the vertices in P_1 and P_2. Its surface (the set of triangles without neighbors) is the convex hull of the vertices.

1.4.3 Third step: tetrahedra elimination

We have to eliminate two kinds of tetrahedra: Those having an edge in P_1 or P_2 outside the contours and those contributing to non-solid connections (see Fig-

ure 1.19). In the graph $\mathcal{G}$, the first kind of tetrahedron is represented by nodes lying on external Voronoi skeletons. These nodes, and the edges of $\mathcal{G}$ adjacent to these nodes are removed from $\mathcal{G}$. *Non-solid connections* are tetrahedra which are only connected along an edge or at one single point to one of the two planes (see Figure 1.19). In the graph $\mathcal{G}$, a t_{12} node is situated on the intersection of

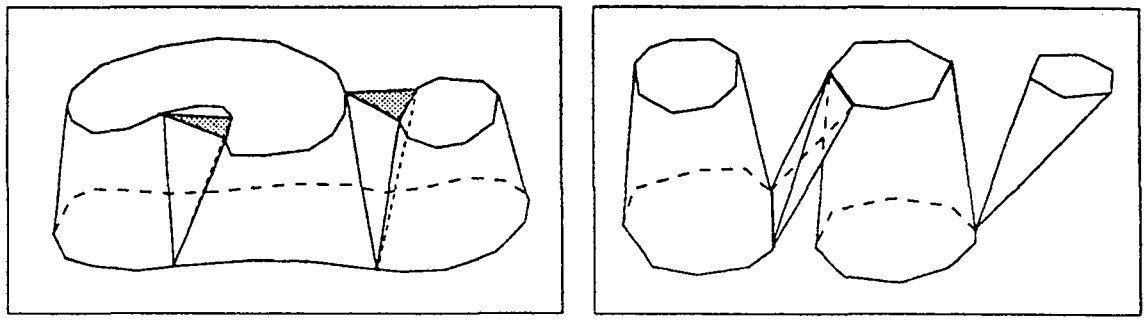


Figure 1.19: Tetrahedra elimination: Tetrahedra outside the contours (left) and non-solid connections (right).

two Voronoi edges, say e_1 belonging to P_1 and e_2 belonging to P_2. A t_{12} node is called *1-solid*, if it is connected across e_1 to at least one of the t_1 nodes on the extremities of e_1. Correspondingly, a t_{12} node is called *2-solid*, if it is connected across e_2 to at least one of the t_2 nodes on the extremities of e_2. The t_{12} nodes that are not i-solid belong to a set of tetrahedra with a one-dimensional connection in P_i, $(i = 1, 2)$ and are deleted from $\mathcal{G}$ (see Figure 1.20). The 0-dimensional connections are represented by connected components of t_1 (t_2) nodes without a connection to a t_{12} node (note that a t_1 tetrahedron cannot be connected directly to a t_2 tetrahedron).

1.4.4 Discussion of results

We can now discuss the behavior of the reconstruction on our graph $\mathcal{G}$. We observe the following facts:

1. The nodes of $\mathcal{G}$ that belong to external Voronoi skeletons have been removed in the elimination steps. We can subsequently remove edges of $\mathcal{G}$ adjacent to eliminated nodes. After this step, $\mathcal{G}$ consists of one or more connected components E_i $(i = 1..k\ ;\ k > 0)$.

2. The edges of $\mathcal{G}$ belonging to internal Voronoi skeletons may be split in several parts by intersecting external Voronoi skeletons. These parts are

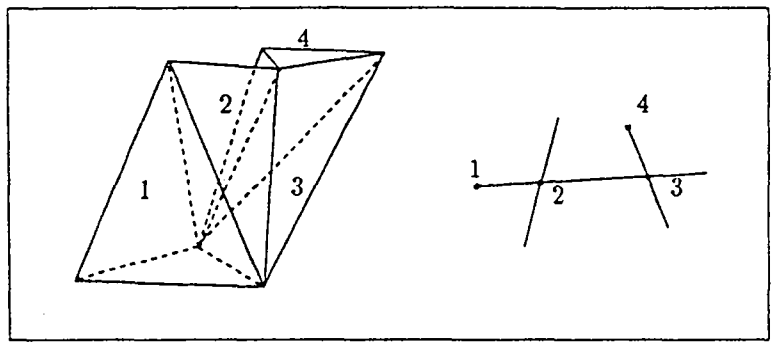


Figure 1.20: The t_{12} tetrahedron 3 is both 1-solid and 2-solid. Tetrahedron 2 is only 1-solid

distributed over some of the connected components E_i.

3. In each connected component E_i, there can be at most one connected part of an internal Voronoi skeleton belonging to P_1 and at most one connected part of an internal Voronoi skeleton belonging to P_2.

4. A contour or a contour part from one cross-section is connected to a contour or a contour part of the adjacent cross-section iff the associated internal Voronoi skeletons or internal Voronoi skeletons parts belong to the same component E_i. This implies that the connected component contains at least one t_{12} node resulting from an intersection between

   - an edge of the internal Voronoi skeleton in P_1 and an edge of the internal Voronoi skeleton in P_2 or
   - an edge of an internal Voronoi skeleton and an interposed Voronoi edges of the other cross-section or
   - two interposed Voronoi edges—one for each cross-section.

5. One single internal Voronoi skeleton in a connected region E_i corresponds to a "dying" contour, that is, a contour which will not be connected to the adjacent cross-section.

6. Triangles in one plane are extended to the vertices which are closest to their circumcenters. The circumcenters are represented by internal Voronoi nodes, and we can guarantee that the internal Voronoi skeleton stays inside the contours. Therefore, contour parts will be connected to their nearest-neighbor contour parts.

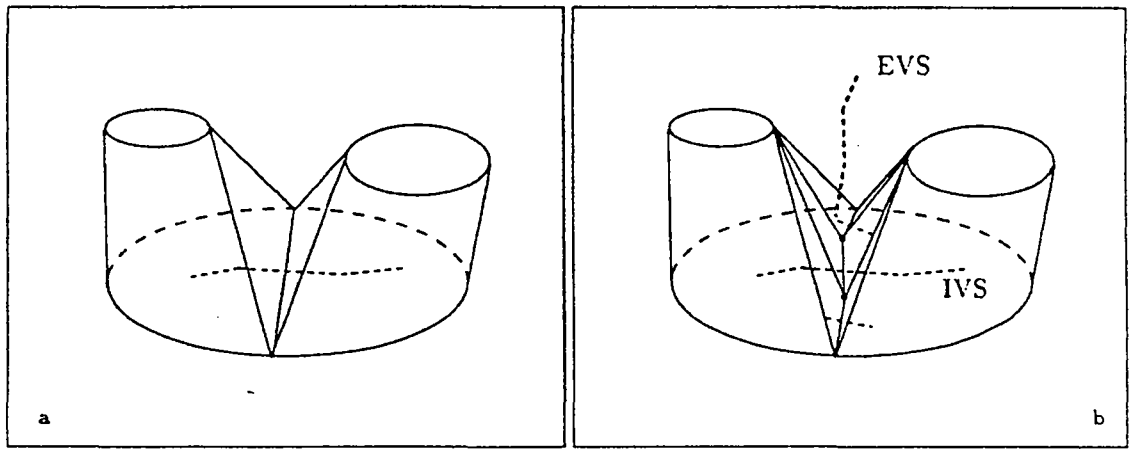


Figure 1.21: Simple branching without internal vertices (left) and our solution (right).

7. If an internal Voronoi skeleton crosses an external Voronoi skeleton of the adjacent cross-section, the associated contour is split along the Delaunay edges corresponding to the intersected edges of the internal Voronoi skeleton. We have shown in the previous section that if we add internal vertices onto the projection of the external Voronoi skeleton, the split line will be close to that external Voronoi skeleton. Hence the branching occurs along an approximation of the locus of equidistance between the branching contours (see Figure 1.21 b).

8. The case of multiple branchings is handled similarly. One contour is divided into several regions by the projection of an external Voronoi skeleton. Each part is connected to one contour in the adjacent plane (see Figure 1.22 b) and Figures 1.24).

9. The separation line in the case of branching contours follows the projection of the external Voronoi skeleton. Figure 1.22 d) shows a prototype of a complicated branching. Figure 1.25 and 1.26 show a simple real example.

10. Due to the addition of internal vertices, the birth and death of holes can be handled without problem (see Figure 1.23). The triangles representing the hole will be connected to the nearest vertices in the adjacent plane. Due to our construction, these are the added vertices.

11. Planar triangles may only be produced inside the regions enclosed by contours, for example at the birth or death of contours. The algorithm never adds horizontal faces outside contours, due to the elimination step.

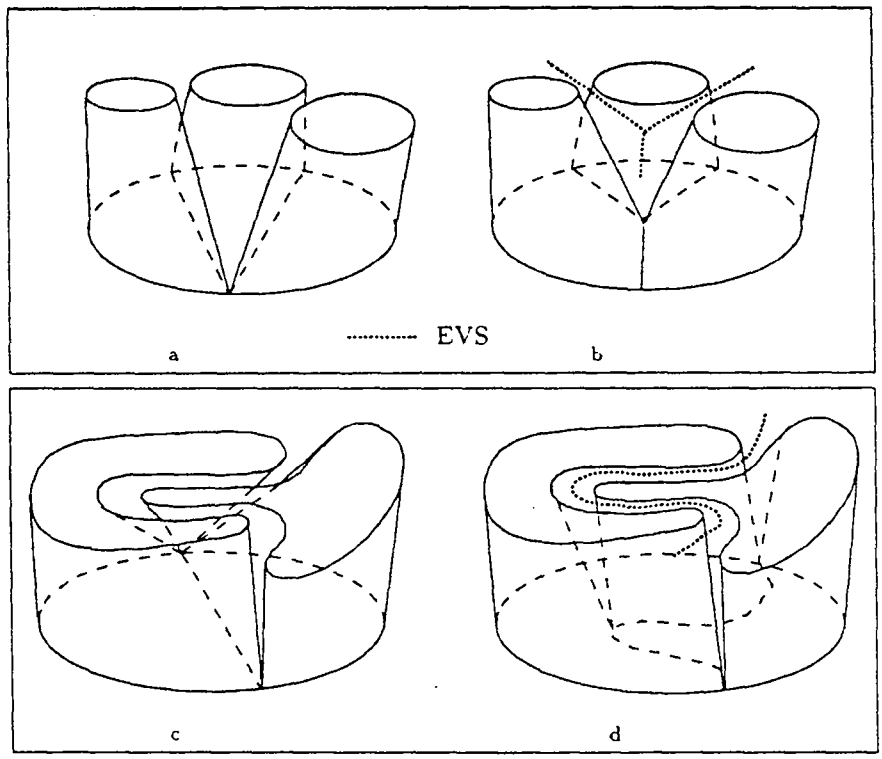


Figure 1.22: Multiple branching (top) and complex branchings (bottom). The examples on the left show the solution without internal vertices, as obtained by the original version of the reconstruction method in [4].

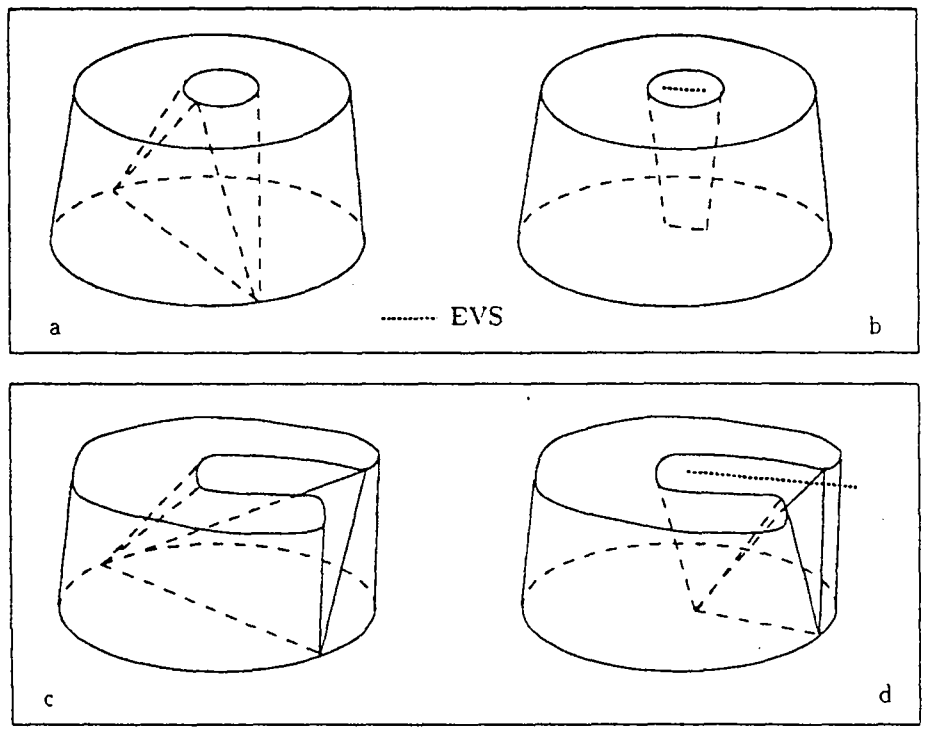


Figure 1.23: Birth of a hole (top) and concave contour (bottom). On the left are the solutions without internal points, yielding double points [4].

12. The factor determining connection is geometric closeness. Multiple contours with large distances are connected to their nearest contours in the adjacent planes, or may split if they are equidistant to contours in the adjacent plane. Figure 1.27 shows the resulting connections of a distant contour. This result clearly differs from the voxel heuristic, where only overlapping regions are connected. This might be desirable behavior, or may be considered a handicap, depending on the application. We propose as an additional parameter to delete all tetrahedra with a slope superior to a given value. The user can so decide if he wants to limit the results strictly to voxel-heuristic, or if he may permit more flexible solutions.

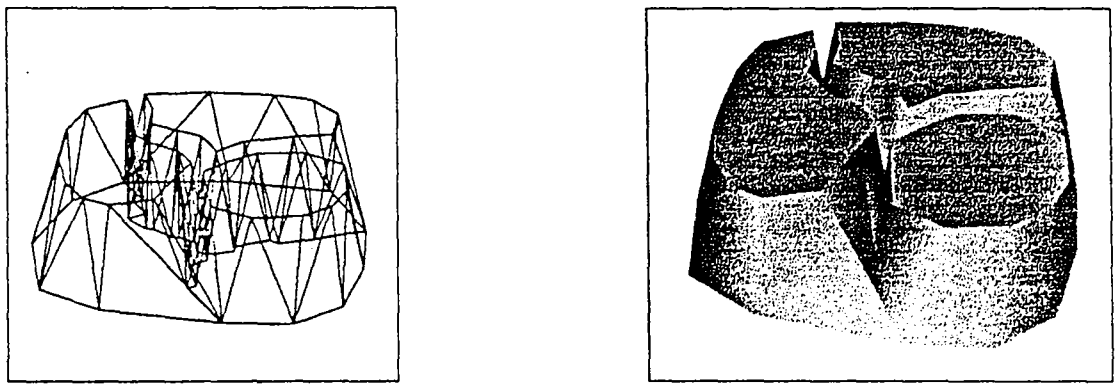


Figure 1.24: An example of splitting one contour into three branches.

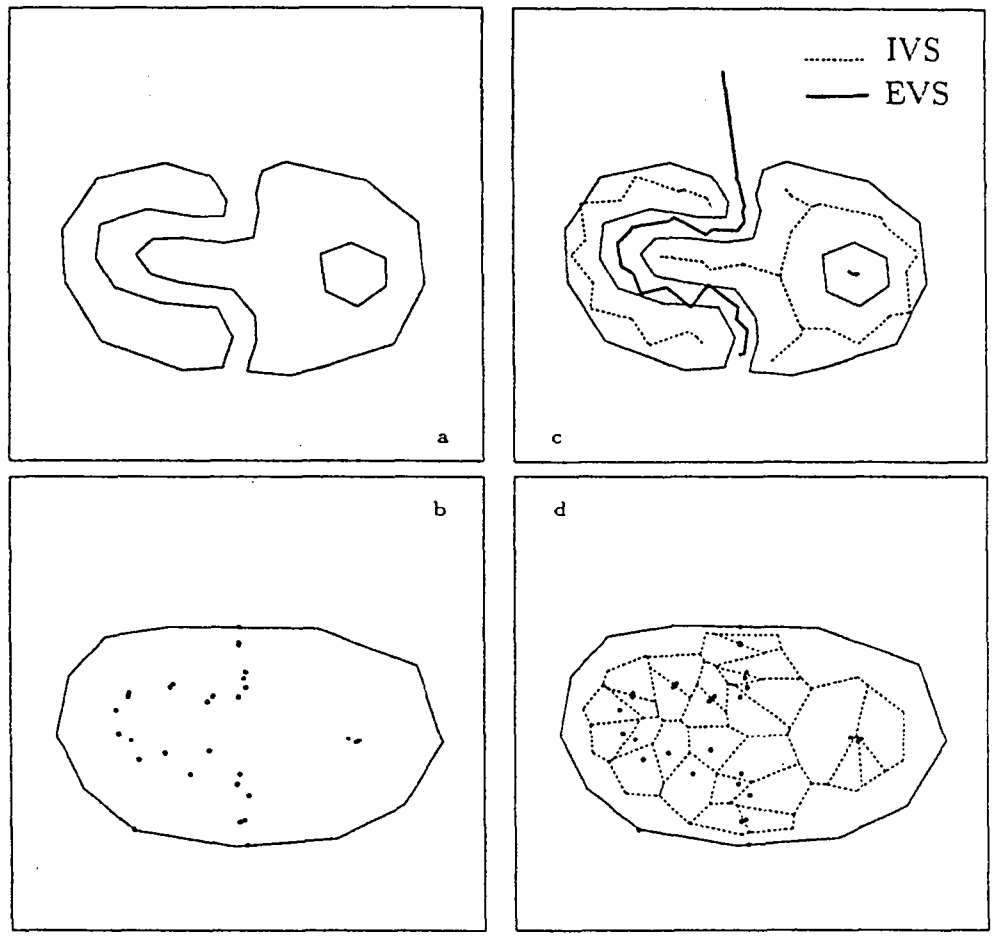


Figure 1.25: An example with complex branching and birth of a hole. a) and b) show two contours in adjacent planes, together with the added vertices. The corresponding internal and external Voronoi skeletons are shown in c) and d).

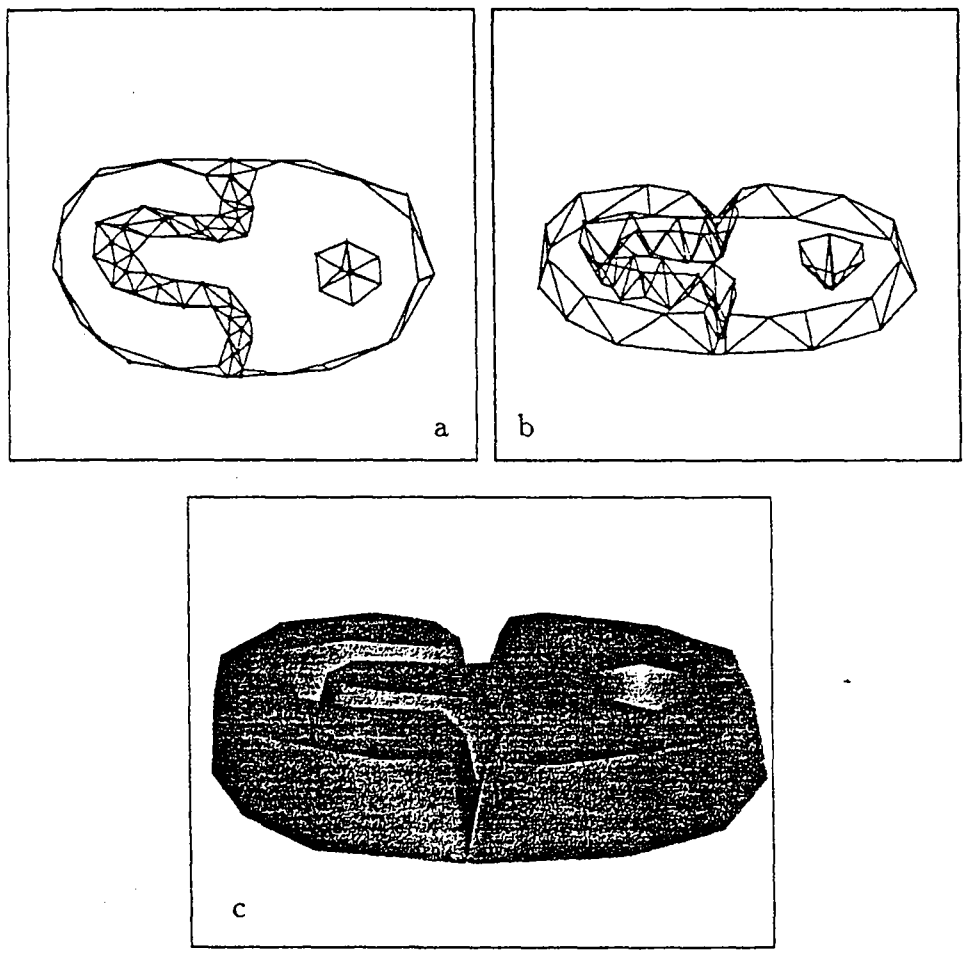


Figure 1.26: The result of the reconstruction of the contours in Figure 1.25. A top view is shown in a). Two perspective views are shown in b) and c).

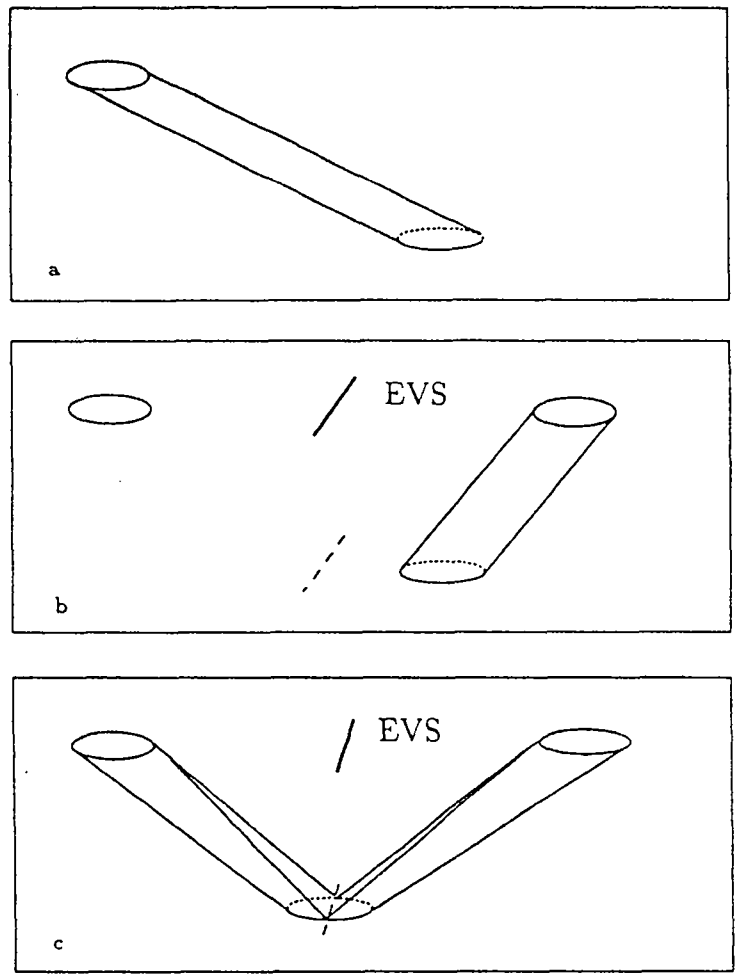


Figure 1.27: A single one-to-one connection (a), a nearest neighbor connection (b) and a split of equidistant contours (c).

Cases which remain unsatisfactory occur with large spaced contours without overlapping :

- Single contours with large distances from one cross-section to another may be connected or not, depending on their shape. This is due to the coarse approximation of the contours. We can show an example of two non-convex contours that will not be connected in Figure 1.28. Because of the concavities, the external Voronoi skeletons make an intersection of interposed Voronoi edges impossible. Two convex contours however will always be connected. Our solution to this dilemma is to increase the number of vertices on the contours depending on some topological information or to delete all tetrahedra with a slope superior to a given limit.

- Single contours without neighbors in adjacent sections will completely disappear in the 3D reconstruction (see Figure 1.29). In this case, the intersection of the reconstruction with the given cross-sections does not contain all original contours.

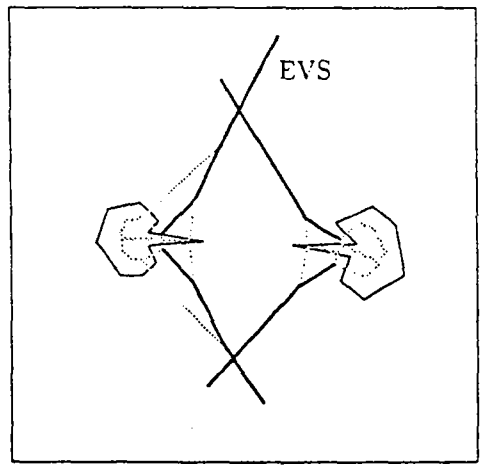


Figure 1.28: The projection of two non-overlapping contours on two adjacent planes that will not be connected. The EVS is drawn in bold

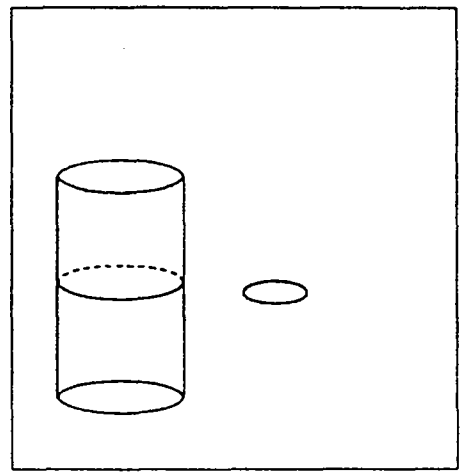


Figure 1.29: A single contour disappears in the final reconstruction.

1.4.5 The complete algorithm

- **ALGORITHM** DELAUNAY_RECONSTRUCTION()
- **INPUT:** A number of contours on parallel cross-sections
- **OUTPUT:** A 3D polygonal reconstruction
  - **For** $i = 0$ **to** number of cross-sections
    - S_i = set of vertices $\in$Plane$_i$
    - C_i = the contour edges $\in$Plane$_i$
    - DT$_i$ = 2D_DELAUNAY(S_i)
    - ADD_VERTICES(DT$_i$, C_i)
  - **For** $i = 0$ **to** number of cross-sections
    - C_i = the contour edges $\in$Plane$_i$
    - ADD_VERTICES_INSIDE(DT$_i$)
    - ADD_VERTICES(DT$_i$, C_i)
  - **For** $i = 0$ **to** number of cross-sections -1
    - $\mathcal{G}$ = 3D_DELAUNAY(DT$_i$,DT$_{i+1}$)
    - REMOVE_TETRAS($\mathcal{G}$)
    - **Output**(tetrahedra, surface triangles)

Since we repeatedly have to add vertices to the Delaunay triangulations, we

use an incremental algorithm to calculate the 2D Delaunay triangulation. We implemented that of Green and Sibson [15].

- **ALGORITHM** 2D_DELAUNAY(S)
- **INPUT:** A set of vertices in a plane
- **OUTPUT:** The Delaunay triangulation of S.
  - ▷ Initialize DT
  - ▷ **For** $i = 0$ **to** number of vertices
    - ▷ INSERT_VERTEX(DT, vertex$_i$)

- **ALGORITHM** INSERT_VERTEX(DT, v)
- **INPUT:** A 2D Delaunay triangulation and a vertex to add
- **OUTPUT:** The Delaunay triangulation with v contained
  - ▷ Find triangle $t \in$ DT containing v
  - ▷ Find the set of triangles T whose circumcircle contains v
  - ▷ Delete T from DT and add new triangles

Adding points on contours to guarantee the contour containment condition may yield a new triangulation with obtuse angles. Adding points to avoid obtuse angles may violate the contour containment. Therefore, we would have to alternate the two procedures until we reach an unvarying configuration. We can give no proof that this algorithm is finite, but we can argue as follows: If we increase the number of vertices on the contours, the Voronoi skeletons get closer to the medial axis (see section 1.3.6). Such a Delaunay triangulation however shows no obtuse angles opposed to contour segments, nor line segments that are not contained.

In our implementation, we limited the number of successive calls. Since the contour containment condition is absolutely required, the last procedure must be CONSTRAIN_SEGMENTS.

- **ALGORITHM** ADD_VERTICES(DT, C)
- **INPUT:** A 2D Delaunay triangulation and a set of contour segments
- **OUTPUT:** A 2D Delaunay triangulation where all segments appear as edges and no obtuse angles are opposed to contour segments
  - ▷ CONSTRAIN_SEGMENTS(DT, C)
  - ▷ DELETE_OBTUSE(DT, C)
  - ▷ CONSTRAIN_SEGMENTS(DT, C)

- **ALGORITHM** CONSTRAIN_SEGMENTS(DT, C)
- **INPUT:** A 2D Delaunay triangulation and a set of contour segments
- **OUTPUT:** A 2D Delaunay triangulation where all segments appear as edges
  - $M = \{0\}$
  - **For** all contour segments $ab \in C$ not in DT
    - Let c be the point in the middle of ab
    - $M = M \cup \{c\}$
    - Delete ab from C and add ac, cb to C
  - **For** all points $c \in M$
    - INSERT_VERTEX(DT, c)
  - **If** $M \neq \{0\}$
    - CONSTRAIN_SEGMENTS(DT, C)

- **ALGORITHM** DELETE_OBTUSE(DT, C)
- **INPUT:** A 2D Delaunay triangulation and a set of contour segments
- **OUTPUT:** A 2D Delaunay triangulation where no obtuse angles are opposed to contour segments
  - $M = \{0\}$
  - **For** all contour segments $ab \in C$
    - Get the two triangles sharing ab as an edge
    - **If** one of the angles opposed to ab is $> \pi/2$
      - Let c be the orthogonal projection of the opposed vertex onto ab
      - $M = M \cup \{c\}$
      - Delete ab from C and add ac, cb to C
  - **For** all points $c \in M$
    - INSERT_VERTEX(DT, c)
  - **If** $M \neq \{0\}$
    - DELETE_OBTUSE(DT, C)

ADD_VERTICES_INSIDE adds only the Voronoi vertices of adjacent external Voronoi skeletons into the contours. The experimental results showed that it is not required to put additional vertices onto the external Voronoi edges.

- **ALGORITHM** ADD_VERTICES_INSIDE(DT)
- **INPUT:** A 2D Delaunay triangulation
- **OUTPUT:** A 2D Delaunay triangulation where vertices have been inserted inside contours
  - $M = \{0\}$
  - **For** each adjacent cross-section A
    - **For** each Voronoi node v on the external Voronoi skeleton of A
      - **If** the projection of v onto DT is inside a contour
        - $M = M \cup \{v\}$
  - **For** all points $c \in M$
    - INSERT_VERTEX(DT, c)

After adding internal vertices, we have to run ADD_VERTICES() once more, since they may contribute to obtuse triangles or missing contour segments.

3D_DELAUNAY calculates the 3D Delaunay triangulation between two adjacent planes.

- **ALGORITHM** 3D_DELAUNAY(DT_1,DT_2)
- **INPUT:** Two 2D Delaunay triangulations in adjacent planes
- **OUTPUT:** A 3D Delaunay triangulation of the vertices $\in (DT_1 \cup DT_2)$
  - **For** each $t \in DT_1$
    - Let v be the vertex $\in DT_2$ which is closest to the circumcenter of t
    - Connect t with v to a tetrahedra of type t_1
  - **For** each $t \in DT_2$
    - Let v be the vertex $\in DT_1$ which is closes to the circumcenter of t
    - Connect t with v to a tetrahedra of type t_2
  - Let $\mathcal{G}$ be the orthogonal projection of the Voronoi diagrams of DT_1 and DT_2
  - **For** each intersection of two Voronoi edges $\in \mathcal{G}$
    - Connect the two corresponding Delaunay edges to a tetrahedra t_{12}
  - Output($\mathcal{G}$)

- **ALGORITHM** REMOVE_TETRAS($\mathcal{G}$)
- **INPUT:** Intersection graph representing a 3D Delaunay triangulation of two adjacent planes
- **OUTPUT:** $\mathcal{G}$ with external tetrahedra and non solid connections removed
  - **For** each node $\in \mathcal{G}$
    - Let t be the tetrahedron represented by that node
    - **If** one of the edges of t lying in one plane is outside of a contour
      - Delete t from $\mathcal{G}$
  - **For** each t_{12} node $\in \mathcal{G}$
    - **If** the node is not 1-solid or is not 2-solid
      - Delete the node from $\mathcal{G}$
  - **For** each t_1 (t_2) node $\in \mathcal{G}$
    - **If** connected component of t_1 (t_2) does not contain a t_{12} node
      - Delete the node from $\mathcal{G}$

1.4.6 Complexity

The number of triangles in a planar Delaunay triangulation is linearly related to the number of vertices. The number of Delaunay edges and Voronoi edges also is $O(n)$, where n is the number of vertices.

Finding the triangle containing a vertex in INSERT_VERTEX() is in our implementation $O(n)$ in the worst case, since we apply a linear search from a starting triangle. However, we can choose this starting triangle close to the next vertex to add, since the vertices are sorted on the contours. So the expected complexity of this step is $O(1)$. The deleting and updating steps have a complexity of $O(k)$, where k is the number of triangles containing the new vertex in its circumcircle. k may be $O(n)$ in the worst case. 2D_DELAUNAY() thus has a worst case complexity of $O(n^2)$, but the expected time increases linearly.

Detecting if all contour edges are contained in a given Delaunay triangulation can be done in $O(n)$ steps. We therefore calculate an adjacency list. For each vertex, we save the adjacent vertices. For each triangle, we thus get 6 entries in the list (3 vertices, each of which has 2 adjacent vertices). The size is therefore $O(n)$. The overall complexity of CONSTRAIN_SEGMENTS() is $O(r\,n)$, where r is the number of times the procedure has to be called. This number is related to the contour complexity, and is very low with realistic data. An upper bound on this problem is given in [11]. The authors show that it is possible to obtain

a Delaunay triangulation of a graph satisfying the containment condition with $O(m^2n)$ vertices, where m is the number of edges and n the number of vertices of the graph.

Detecting all obtuse triangles can be done in $O(n)$ steps. We use a data structure, which contains for each vertex the triangles sharing it. Each triangle adds 3 entries in this list, so the complexity of DELETE_OBTUSE() is $O(ln)$, where l is the number of recursive calls. Again, this number depends on the contour complexity and is reasonably small in practice.

The number of vertices which are added in ADD_VERTEX_INSIDE() depends very much on the contour shapes and on the shape difference between adjacent cross-sections. The complexity of that procedure is related to the number of vertices on the external Voronoi skeleton, which may be of order $O(n)$ in the worst case. To decide if a point lies inside or outside of a contour is done in $O(n)$ worst case complexity, since we use the same triangle search as in INSERT_VERTEX(). Again we take the advantage of a relatively ordered distribution of the Delaunay triangles, since the vertices have been inserted sorted on contours.

This same property of our Delaunay triangulation is useful to rapidly find the nearest vertices in order to build the t_1 and t_2 tetrahedra. Our implemented algorithm has a worst case complexity of $O(n)$ per triangle, but runs in constant time in practical cases.

The construction of t_{12} tetrahedra requires the calculation of the intersections between two Voronoi diagrams projected onto the same plane. To find the intersections on one Voronoi edge, we have to cross a number of Voronoi cells. That number corresponds to the number of intersection on the edge. The starting and ending cell have been calculated in the preceding step, when constructing the t_1 and t_2 tetrahedra. With an appropriate data-structure, we can rapidly change from one cell to an adjacent cell. Thus, finding the intersection of the Voronoi edges of two adjacent planes is done in $O(K)$ time, where K is the number of intersections.

In the worst case, K can be $O(n^2)$. This theoretical result was encountered in an experiment, where the contours were filled with a number of equally spaced vertices. This emphasizes the need of thinning pixel-contours (contours where each pixel contributes to a vertex) by at least deleting colinear vertices (see also the following chapter). With realistic contour shapes, K is nearly linearly related to the number of vertices.

In the elimination step, we have to visit each tetrahedron to decide if it is inside or outside of contours. Detecting the non-solid-connections requires $O(K)$ time, as shown in [4]. The overall complexity of the Delaunay reconstruction yields $O(n^2 + K + ln + rn)$ in the worst case, with linear expected time for data sorted on contours.

1.5 Experimental Results

The algorithm has been implemented in C. The program called NUAGES contains four tools:

- the preprocessor *prepros*
- the reconstruction part *repros*
- the postprocessing parts *repros2visu* and
- the postprocessing part *repros2tetra*.

The preprocessor has the following features:

- test for intersections or self-intersections of contours.
- correction of contour orientation.
- deletes incident vertices.
- polygonal approximation of pixel contours with a given error to reduce the number of vertices on the contours. This is especially useful for automatically segmented contours (optional). The procedure is described in the following chapter.
- perturbs identical vertices in adjacent cross-sections (optional).

Numerical problems arise during the reconstruction procedure *repros*. Since the input data is on a regular grid, we frequently get more than three vertices on a common circle, and more than four vertices on a common sphere, especially with automatically segmented contours. The first case is a problem when constructing the Delaunay triangulation, the second makes it difficult to calculate the intersections of two adjacent Voronoi diagrams. This problem is partially solved by reducing vertices in a contour approximation. Persisting co-cyclic vertices are perturbed during the insertion procedure.

The postprocessor *repros2visu* is used for the computation of surface triangles. It calculates the interpolated surface normals in each vertex. The surface normal n_v in vertex v is the sum of the normals of all triangles sharing v as a vertex, weighted with the angle at v:

$$n'_v = \sum_{i=1}^{k} n_i \alpha_i \quad \text{and} \quad n_v = \frac{n'_v}{\|n'_v\|}$$

where n_i is the normalized normal of triangle i and α_i is the angle between the two triangle edges joining in v. Optionally, we can select cross-sections which will not be closed by horizontal faces or which will not be smoothed.

Repros2tetra is used to process the tetrahedra output. It completes the neighbor links between adjacent slices, and labels connected components.

We tested the program on various medical data:

- The brain data has been acquired and segmented by Surgical Planning Laboratory, Department of Radiology, Brigham and Women's Hospital and Harvard Medical School [2]. It is from a set of 123 MR images. The contour extraction was automatic, with minimal human assistance. Cross-section distance is 1.5mm, and pixel to pixel distance is 1mm.

- The pelvis data is provided from Resonex Inc. The in-plane resolution is 1.1mm, the cross-section distance is 8.625mm. The bone contours from the 23 cross-sections were extracted manually.

- The knee data is from the Chapel Hill Volume Rendering Test Dataset, Volume I [3]. The contours are drawn manually.

- The heart and lungs were extracted manually from a MR data set provided from the Institut of Biokybernetik und Biomedizinische Messtechnik at the Universität Karlsruhe. Slice distance was about 5mm.

- The head data and femur data are from the Universitaetsklinik Heidelberg. The slice distance was 8mm, and the contour extraction was done manually.

In all practical cases, the CPU time is almost linearly related to the number of contour vertices, if we take the added vertices into account. The number of added vertices depends on the contour complexity. The simplest contour set is that of the head, where the number of added vertices was 18% of the input vertices. The brain dataset shows a rate of 57% added points because of its great number of concave parts and internal contours. The CPU time on a DECstation 5000 is less than 5ms per input vertex. Table 1.1 shows some computational results in detail. Figures 1.30 to 1.38 show reconstructions from the pelvis data and the brain data.

[2]Contact: Ron Kikinis, M.D. tel (617) 732-5961 email kikinis@bwh.harvard.edu

[3]Available via anonymous ftp at cs.unc.edu

	cross sections	contours	points	added points	cpu time	time/ point	surface tiles	tetra-hedra
pelvis	23	84	2059	622	6.50s	2.42ms	5518	16169
knee	42	106	2028	191	5.33s	2.40ms	4386	11783
heart	29	58	1222	243	3.60s	2.45ms	2988	6388
lungs	34	86	3034	447	8.58s	2.46ms	7016	19525
femur	14	38	926	244	2.68s	2.29ms	2396	5292
head	17	25	856	155	2.35s	2.32ms	2032	5137
brain	111	398	23285	13196	107.86s	2.94ms	114028	237192
white	109	520	30191	7218	94.63s	2.53ms	74489	235637
csf	69	235	2536	653	7.75s	2.43ms	5682	15946

Table 1.1: Some experimental results

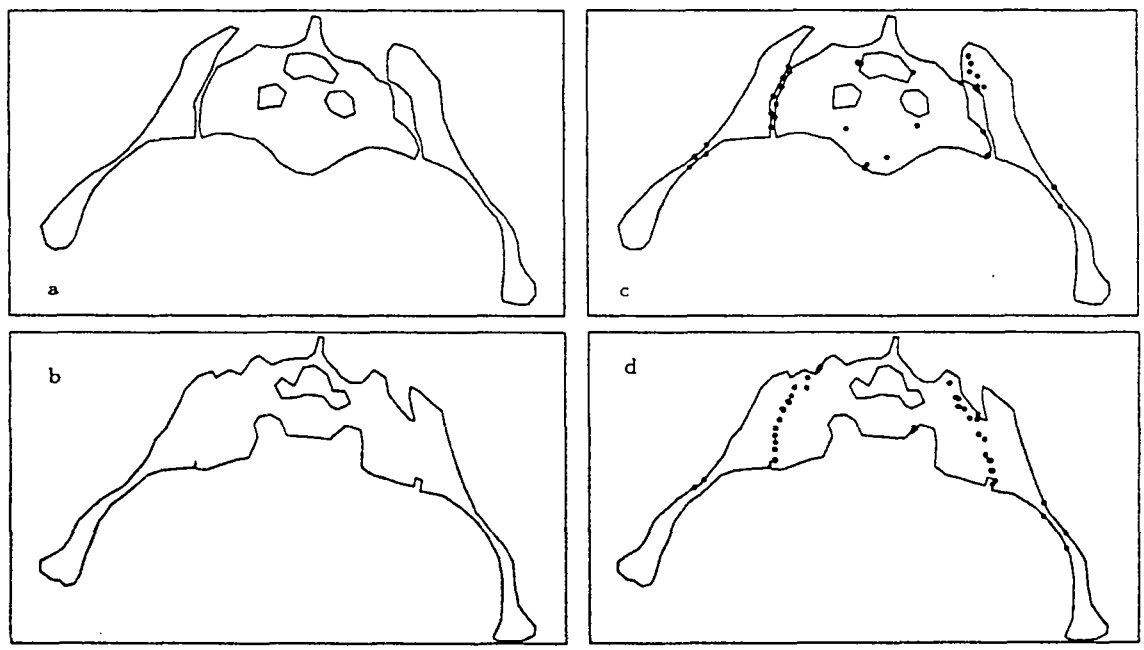


Figure 1.30: a) and b) show two adjacent cross-sections of a pelvis. Slice distance is about 8mm. The vertices added during the reconstruction are shown in c) and d).

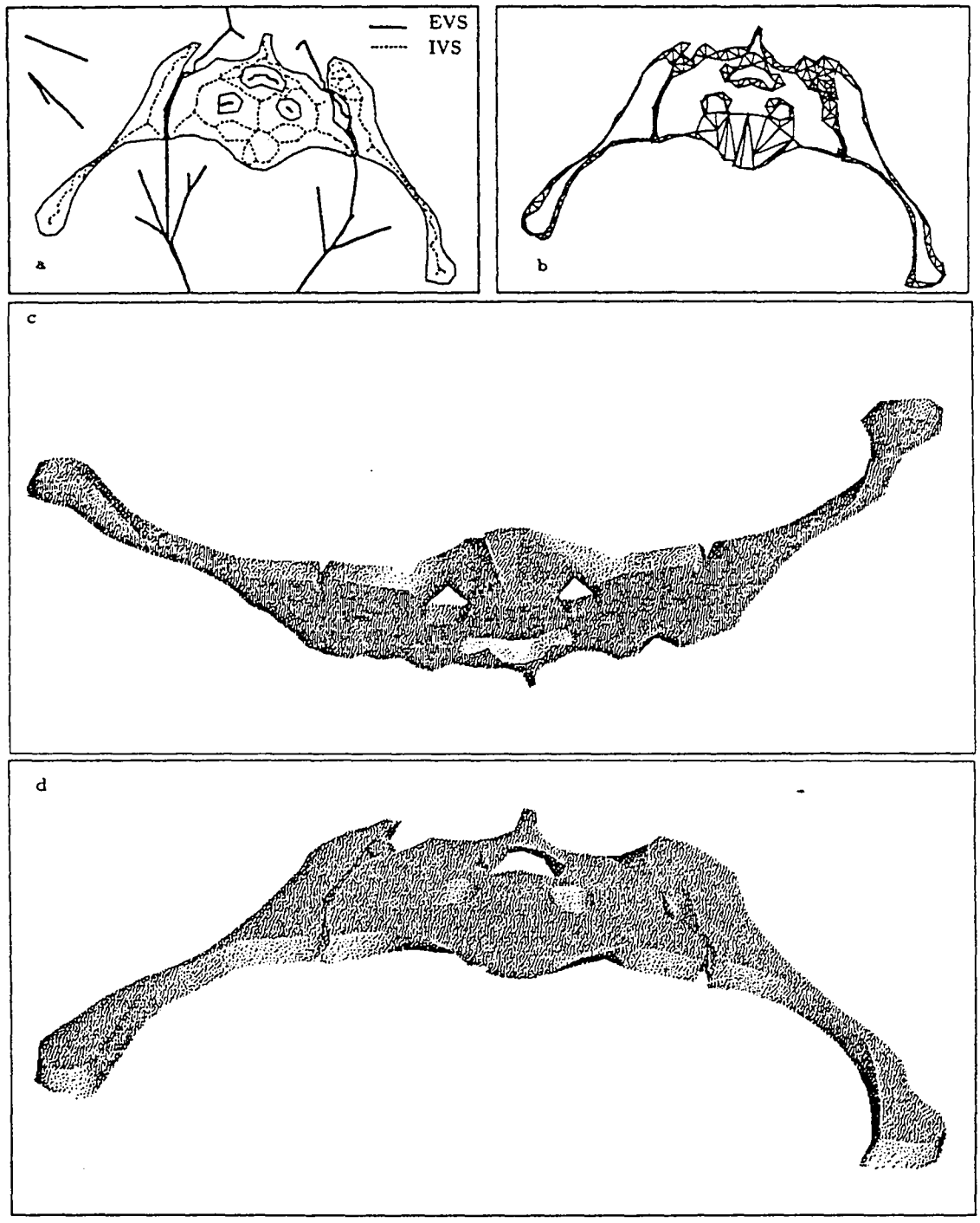


Figure 1.31: a) The upper cross-section of Figure. 1.30 with internal (bold) and external (bold dashed) Voronoi skeleton. b) The reconstructed surface between the two sections, seen from above. c) and d) are two shaded perspective views of the slice.

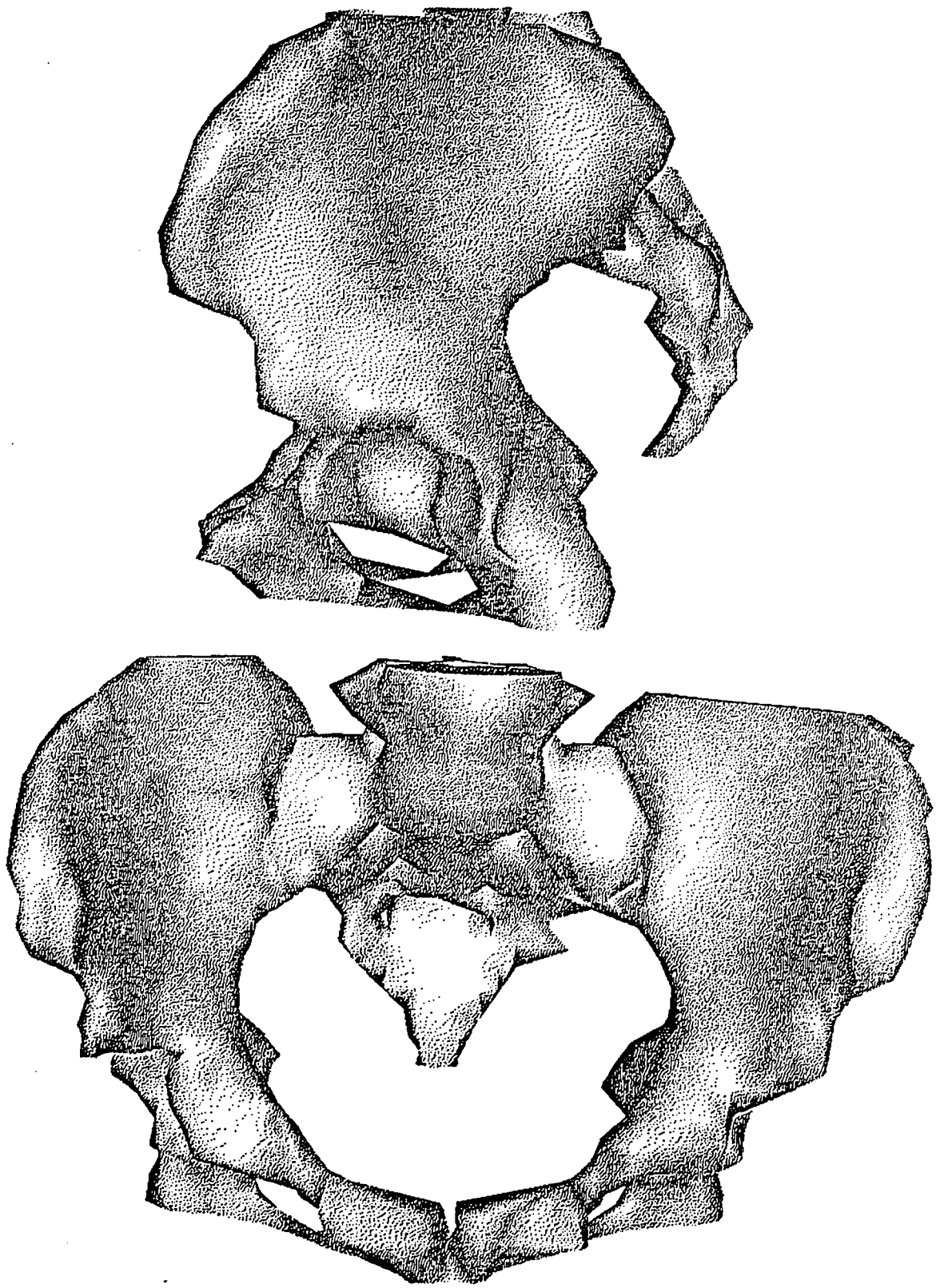


Figure 1.32: The overall reconstruction of a human pelvis from 23 *MR* images.

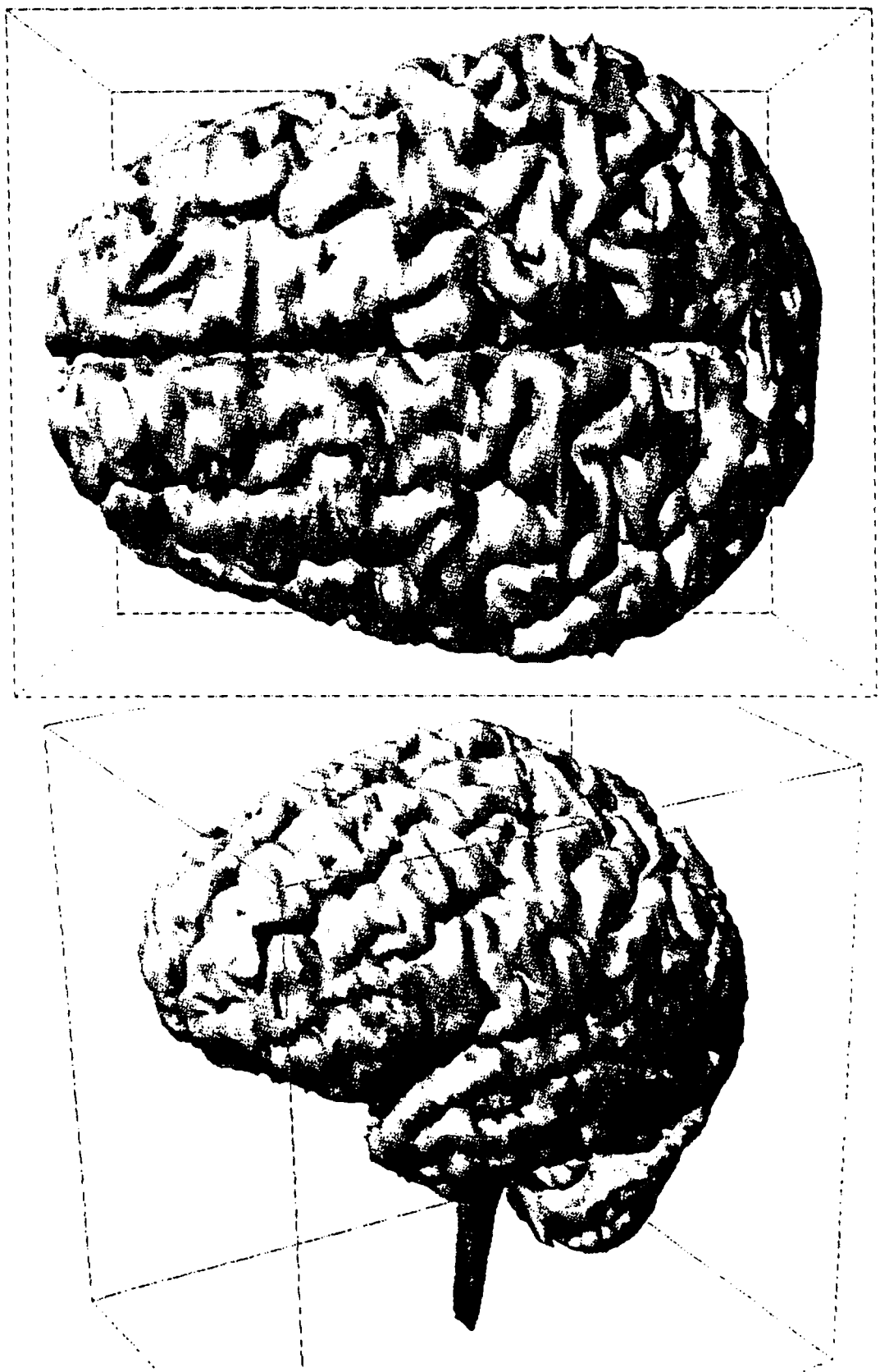


Figure 1.33: A human brain from 123 pre-segmented MR images.

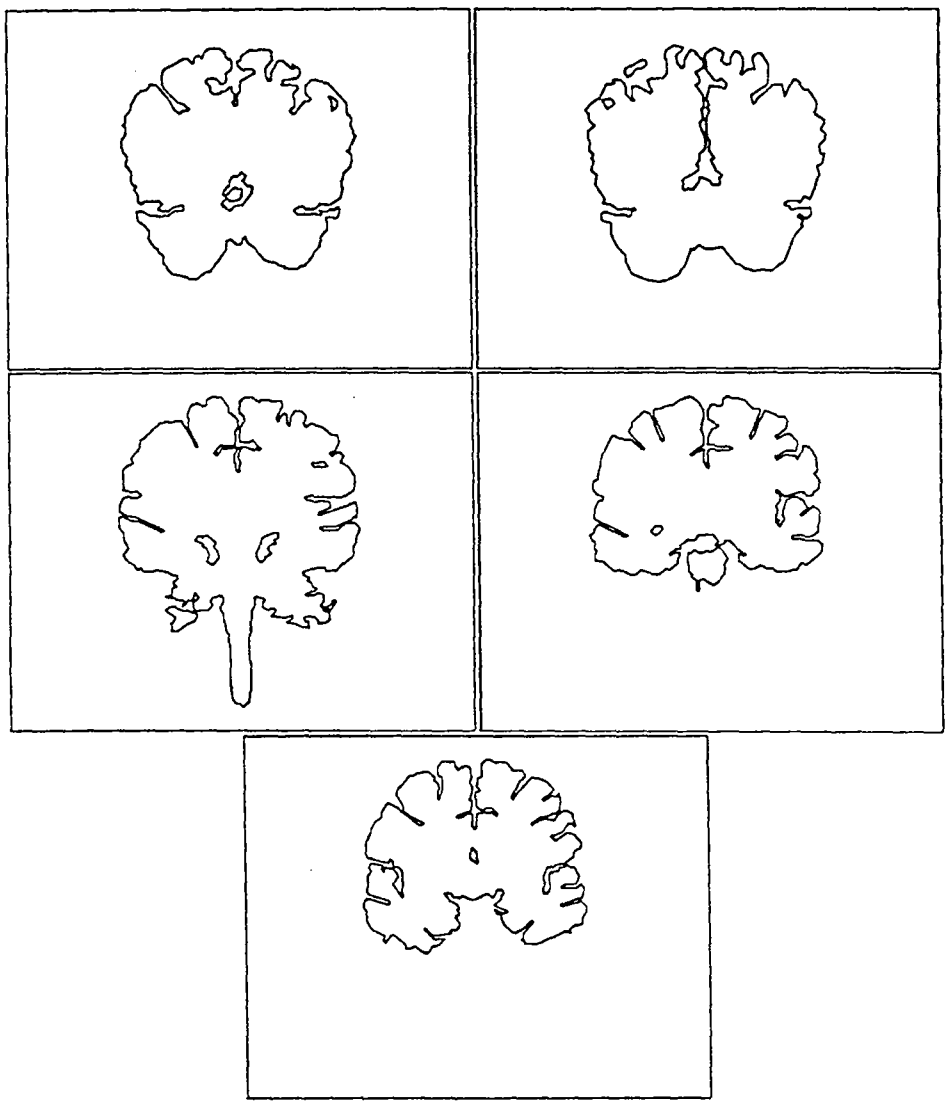


Figure 1.34: Some contours from the brain dataset.

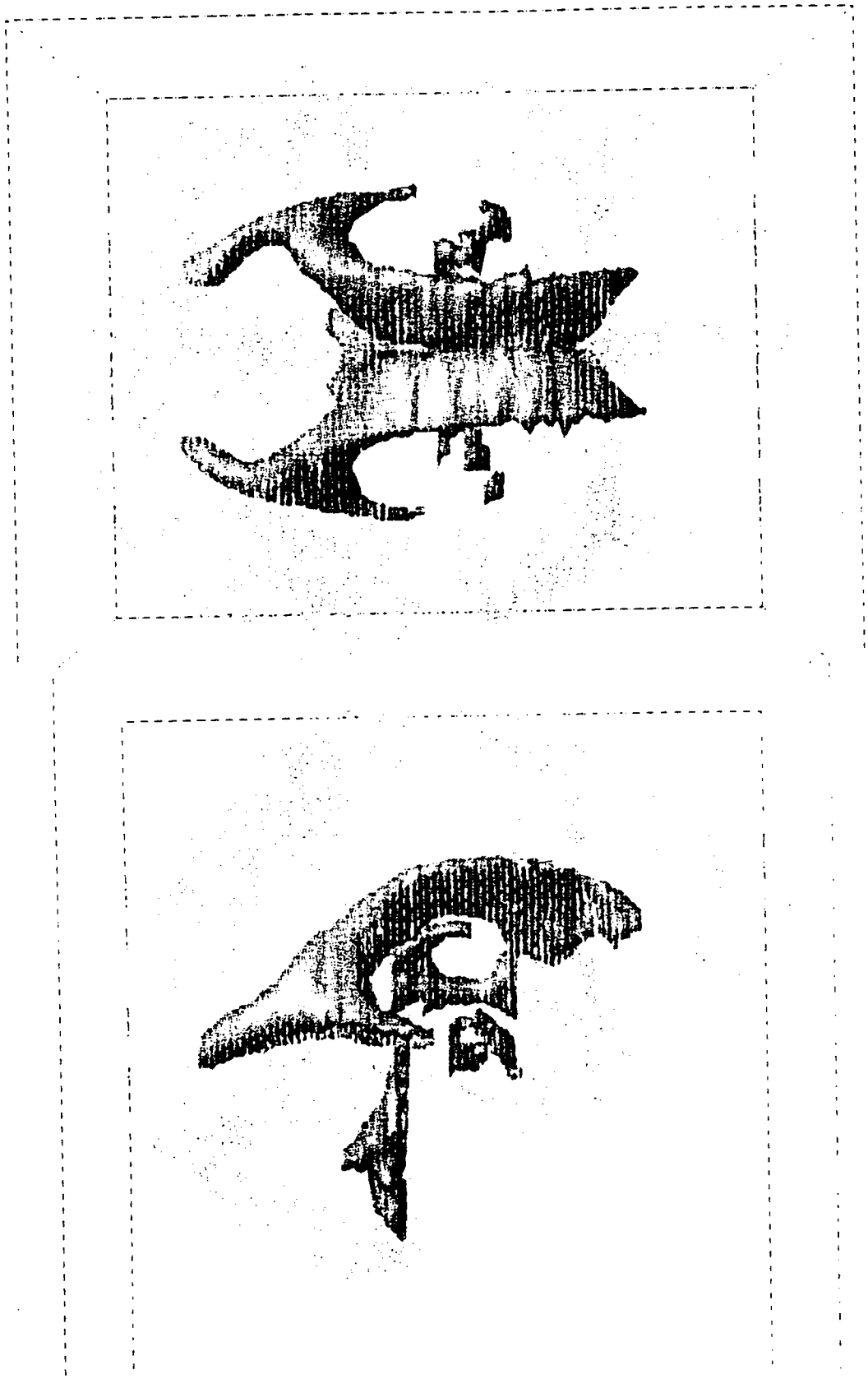


Figure 1.35: The cerebrospinal fluid (CSF) from the brain data set.

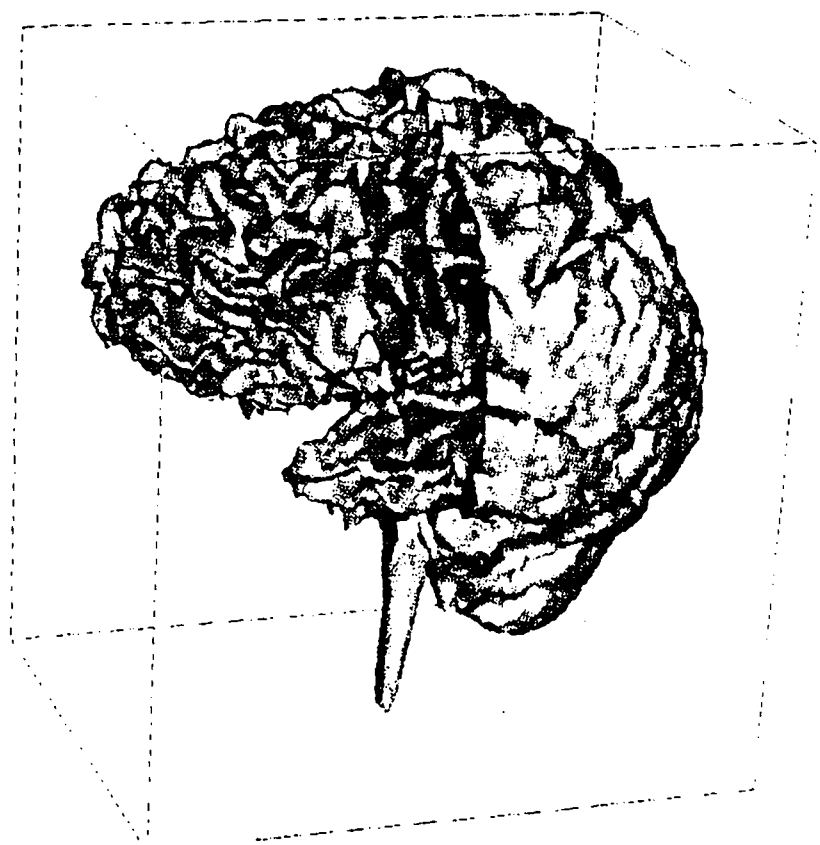


Figure 1.36: Gray matter (outside) and white matter (inside)

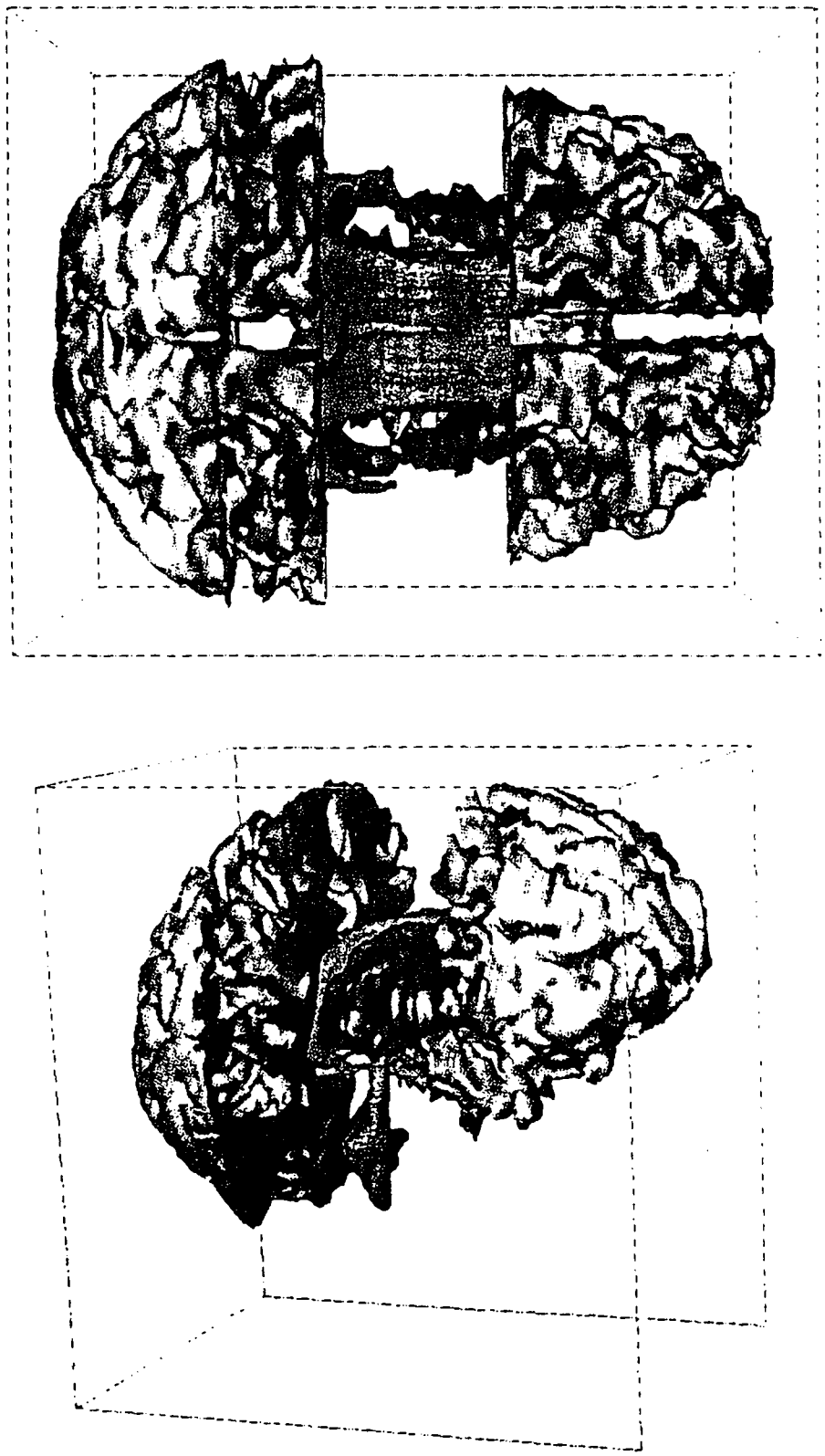


Figure 1.37: Gray and white matter, CSF (red), caudate nucleus (blue), putamen (green), thalamus (yellow)

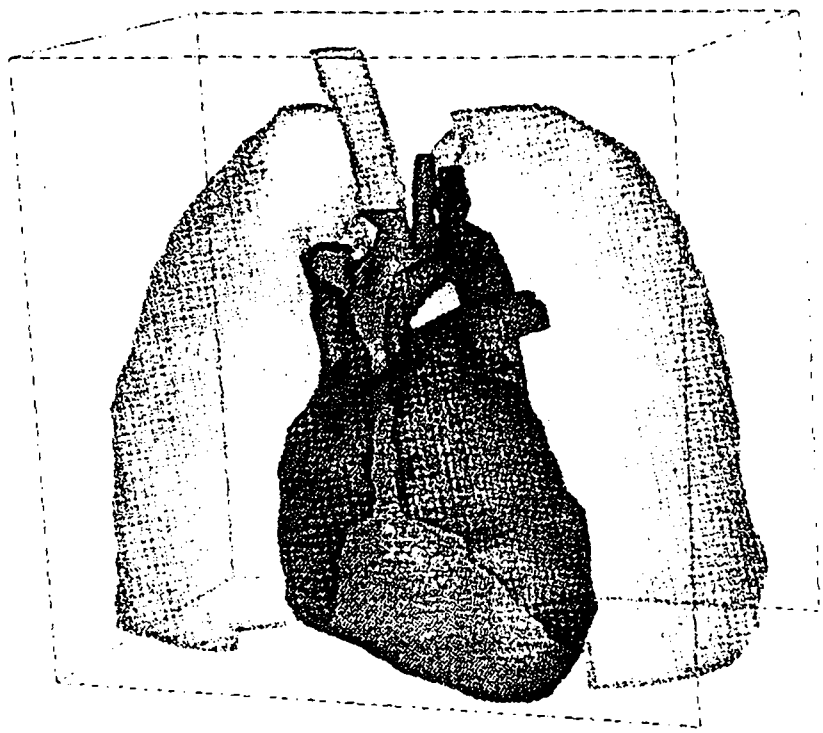


Figure 1.38: Heart and lungs.

1.6 Conclusion

We have demonstrated a solution to the reconstruction problem using the 3D Delaunay tetrahedrization. Our method is based on geometric closeness. This is a simple heuristic, similar to that of the voxel technique. We connect the regions which are intersecting in the orthogonal projection. However, in addition to connecting overlapping contour parts vertically, we also connect non-overlapping parts to their nearest neighbors. We have shown how the Voronoi diagram and Delaunay triangulation can be used to obtain an approximation of an "exact" nearest neighbor connection. The quality of the approximation is adjustable by automatically adding vertices onto and inside the contours. Our method cannot solve specific constraints, in particular we cannot force connections which do not correspond to geometric closeness. However, for a general reconstruction purpose it is a reliable and powerful tool.
The method has several properties:

- It gets directly to a 3D polyhedral representation composed of tetrahedra.
- The property of connecting contours on adjacent planes directly by triangles avoids the need for anti-aliasing or interpolation steps, especially for large cross-section distances.
- Complex contours with multiple branchings, birth and death of holes and complicated splitting lines are handled correctly.
- We get a considerable data reduction compared to other volume oriented methods. Real time display of reconstructed human organs is therefore possible on standard graphic workstations. This feature may be interesting for the design of models used in virtual reality, where rendering speed is crucial.
- The tetrahedral structure can be used for applications like simulation of motion or finite element methods (see Chapter 3).

For the future, we are interested in the following problems:

1. We have shown how to calculate a 3D Delaunay triangulation from two 2D Delaunay triangulations on parallel planes. We added points to get an approximation of the medial axis. Would it be possible to calculate a 3D "region" Delaunay triangulation from two 2D edge Delaunay triangulations on two parallel planes, by eventually adding generalized Voronoi edges inside contours ?

2. Each reconstructed slice is a "constrained" 3D Delaunay triangulation of vertices. It is constrained, because the contour segments are contained. The overall 3D reconstruction however is not a Delaunay triangulation, because adjacent cross-sections may violate the empty sphere property. We would like to calculate an overall 3D Delaunay triangulation under the constraint that the same surface triangles and contour segments are contained. Such a structure would be of interest for 3D shape representation.

3. The widespread use of ultrasound yields an interest in developing methods that work on arbitrarily oriented cross-section. The construction of a 3D Delaunay triangulation of points distributed on non-parallel planes is described in [5, 6]. Maybe this technique could also be applied successfully to the reconstruction problem.

Chapter 2

Verification of Accuracy

2.1 Introduction

We showed in the previous section how to build 3D models from cross-sectional contours. Depending on the application, these models have to be more or less accurate. In cases where the reconstruction is done to visualize the spatial relationship between objects, accuracy plays a less important role. For medical applications however, accuracy may well be vital. We shall mention just two examples: radiation therapy planning, where inaccurate models may result in the irradiation of healthy tissue; and stereotactic surgical planning, where a specific locus in the brain has to be reached precisely. In this chapter, we shall explain our methods of calculating the precision of the Delaunay reconstruction, and show the results on synthetic cross-section data. We shall also compare our technique to the marching cube method.

In order to obtain a 3D model, we must carry out three processes:

- Data acquisition on a Scanner,
- Contour extraction and
- 3D reconstruction.

Each of these steps adds an error to the final result. There are several possible methods of calculating the error from each step. The most natural way is to let a real object undergo the three procedures, and after each step, compare the results to reality.

The accuracy of the image acquisition step depends upon the imaging technique (MRI, X-Ray), the scanner types, and the image resolution. Our pelvis images, for example, are corrected to less than 2 percent geometric distortion. We will not investigate this part any further but will concentrate on edge extraction and reconstruction.

We generate synthetic cross-sectional images from an algebraic object. The contours are extracted both manually and automatically, and undergo the reconstruction procedure. The resulting polyhedron is compared to the algebraic object. In the following section, we shall discuss the measurements necessary to assess the "resemblance" of 3D models. We shall then present our test-data generator and the test program. In the last section we shall discuss the results. We shall only give brief sketches of the algorithms. They are not optimal but are easy to implement, and the execution time is acceptable.

2.2 Definition of similarity measure

Given an algebraic object $\mathcal{A}$ in 3D space, and a polyhedron $\mathcal{P}$ such that $\mathcal{P}$ is the approximation of $\mathcal{A}$. How can we define and measure the resemblance between $\mathcal{A}$ and $\mathcal{P}$?

Comparing the volumes has little significance. Missing parts at convex sides may compensate with enlarged portions at concave sides. For the same reason, calculating the surface also gives a very rough approximation. The difference volume

$$\delta V = (A \cup P) \setminus (A \cap P)$$

gives a better indication of shape difference. Another important measure is the Hausdorff distance. The Hausdorff distance δH of $\mathcal{A}$ and $\mathcal{P}$ is given by

$$\delta H(\mathcal{A}, \mathcal{P}) = max\left(\sup_{a \in \mathcal{A}}(\inf_{p \in \mathcal{P}}(d(a,p))), \sup_{p \in \mathcal{P}}(\inf_{a \in \mathcal{A}}(d(a,p)))\right)$$

where $d(a,p)$ denotes the euclidean distance between a and p. In other words, the Hausdorff distance is the maximum of the maximal distance from a point of $\mathcal{A}$ to $\mathcal{P}$ and the maximal distance from a point of $\mathcal{P}$ to $\mathcal{A}$.

The Hausdorff distance and the difference volume are two complementary measures: The Hausdorff distance indicates the locally "worst" point, but it does not tell how often this or similar cases occur. The difference volume gives a more global value of resemblance, but no worst cases are reported. Objects with equal difference volume may have a divergent Hausdorff distance, and vice versa (see Figure 2.1).

Comparing the surface normals is important for two reasons: Firstly, they determine the visual quality when the object is displayed in Gouraud or Phong shading. Secondly, they are used in motion planning applications, where forces are deduced from surface normals at contact points. Given two points $p \in \mathcal{P}$ and $a_0 \in \mathcal{A}$, with $d(p,a_0) = \inf_{a \in \mathcal{A}}(d(p,a))$. This means that a_0 is the corresponding nearest point to p. We define the *normal difference* $\alpha(p,\mathcal{A}) = \arccos(n(p) \cdot n(a_0))$,

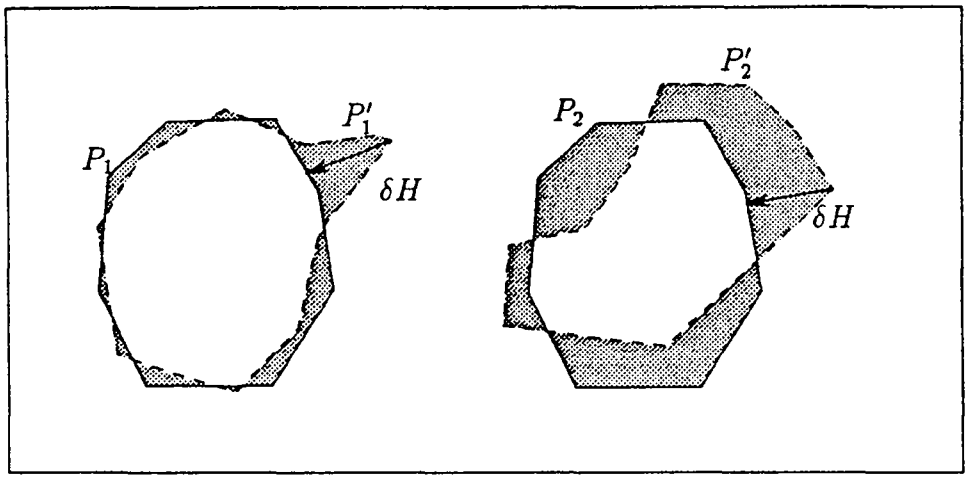


Figure 2.1: Two pairs of polygons. The Hausdorff distance of P_1 and P_1' is the same as of P_2 and P_2'. The difference volume (gray) is greater for P_2 and P_2'.

where $n(a_0)$ is the surface normal of $\mathcal{A}$ in point a_0, and $n(p)$ is the reconstructed surface normal of $\mathcal{P}$ in point p. Similar to the Hausdorff distance, we can define the largest normal distance as

$$\delta N = max\left(\sup_{a\in\mathcal{A}}(\alpha(a,\mathcal{P})), \sup_{p\in\mathcal{P}}(\alpha(p,\mathcal{A}))\right)$$

2.3 Synthetic cross-section generation

As a synthetic object, we selected a torus for several reasons. Firstly, its cross-sectional contours represent convex and concave portions and even holes, depending on its slope relative to the cutting planes. Secondly, its algebraic form is simple. A point t on the torus surface $\mathcal{T}$ is given by

$$t(\varphi,\theta) = \begin{pmatrix} (R + r\ \cos\theta)\ \cos\varphi \\ r\ \sin\theta \\ -(R + r\ \cos\theta)\ \sin\varphi \end{pmatrix}, \quad 0 \le \varphi, \theta < 2\pi.$$

The distance $d(p,\mathcal{T})$ of a given point p to the torus surface can be determined by

- calculating φ: it is the angle between the plane $C_{z=0}$ and the plane parallel to the y axis and passing through O and p. With φ, we get the point $c(\varphi) = (R\cos\varphi, 0, -R\sin\varphi)$ which is the center of the small torus disk.

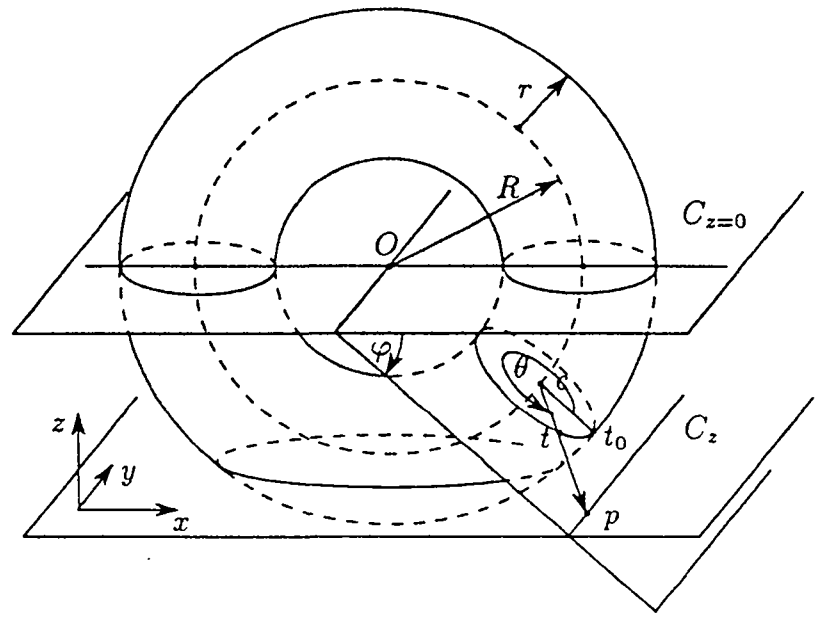


Figure 2.2

- calculating θ: it is the angle between the edges $\overline{c(\varphi), t(\varphi, 0)}$ and $\overline{c(\varphi), p}$.
- $d(p, \mathcal{T})$ is then given by the distance $d(t(\varphi, \theta), p)$.

Furthermore, we define the $Sign(d(p, \mathcal{T}))$ negative, if p is inside $\mathcal{T}$ and positive otherwise. The distance $|d(p, \mathcal{T})|$ is equivalent to $\inf_{t \in \mathcal{T}}(d(p, t))$.

Another advantage in using a torus is the fact that for calculating the Hausdorff distance, it is sufficient to evaluate the maximal distance from a point of $\mathcal{P}$ to $\mathcal{T}$, as we show in the following paragraph:

CLAIM: *Given a torus $\mathcal{T}(R, r)$ with main radius R and small radius r, and a polyhedron $\mathcal{P}$ with triangular faces. We express the fact that $\mathcal{P}$ is an approximation of $\mathcal{T}$ by the conditions*

i All surface triangles of $\mathcal{P}$ are contained within the two tori $\mathcal{T}(R, 1/2\,r)$ and $\mathcal{T}(R, 3/2\,r)$ and

ii The main circle with radius R is always inside $\mathcal{P}$.

Then for each $t_0 \in \mathcal{T}$ there exists at least one $p(t_0) \in \mathcal{P}$ so that $d(p(t_0), t_0) = \inf_{t \in \mathcal{T}}(d(p(t_0), t))$, and

$$\sup_{t \in \mathcal{T}}(\inf_{p \in \mathcal{P}}(d(t, p))) \leq \sup_{p \in \mathcal{P}}(\inf_{t \in \mathcal{T}}(d(p, t))).$$

PROOF: For a given point $t_0(\varphi, \theta) \in \mathcal{T}$, we draw a half line starting in $c(\varphi) = (R\cos\varphi, 0, -R\sin\varphi)$ and passing through t_0. Because of (ii), this line intersects $\mathcal{P}$ in at least one point. Let $p(t_0)$ be the first intersection point. Then $d(p(t_0), t_0) = \inf_{t\in\mathcal{T}}(d(p(t_0), t))$.
Now given two points $t_0 \in \mathcal{T}$ and $p_0 \in \mathcal{P}$ with

$$d(t_0, p_0) = \sup_{t\in\mathcal{T}}(\inf_{p\in\mathcal{P}}(d(t, p))).$$

Since p_0 is the nearest point to t_0, we can also write

$$d(t_0, p_0) = \inf_{p\in\mathcal{P}}(d(t_0, p))$$

If we replace p_0 by another point on $\mathcal{P}$, $p(t_0)$, we get:

$$d(t_0, p_0) \leq d(p(t_0), t_0).$$

Since $p(t_0)$ is the point on $\mathcal{P}$ for which $d(p(t_0), t_0) = \inf_{t\in\mathcal{T}}(d(p(t_0), t))$, it follows that

$$\sup_{t\in\mathcal{T}}(\inf_{p\in\mathcal{P}}(d(t, p))) \leq \sup_{p\in\mathcal{P}}(\inf_{t\in\mathcal{T}}(d(p, t))).$$

□

The method used to produce cross-sectional images of the torus is quite simple. The distance of each pixel p_{ij} to $\mathcal{T}$ is calculated. The density value is set depending on that result:

$$\text{density}(i,j) = \begin{cases} 255 & \text{if } d(p_{ij}, \mathcal{T}) < -0.5 \\ 0 & \text{if } d(p_{ij}, \mathcal{T}) > 0.5 \\ (d(p_{ij}, \mathcal{T}) - 0.5) * 255 & \text{otherwise} \end{cases}$$

This reflects the fact that density values are mean values over a certain pixel area. The image size is 256×256 pixels. The pixel center coordinates are defined from -127.5 to $+127.5$ in both x- and y-directions. The z-coordinate is variable and determined by the height of the cross-section. Additionally, the torus may be rotated around the x axis to generate more complex topologies.

- **ALGORITHM** CROSS_SECTION_IMAGE_GENERATOR
- **INPUT:** a rotation angle α and a number of z-values
- **OUTPUT:** A number of 256 × 256 images representing cross-sections of a torus at a slope of α degrees
  - ▷ **For each** z-value z_i
    - ▷ **For each** pixel p_{ij} in the plane $C_{z=z_i}$
      - ▷ rotate the coordinates of p_{ij} by $-\alpha$
      - ▷ calculate $d(p_{ij}, T)$
      - ▷ calculate density of p_{ij}
  - ▷ output image

The cross-sectional contours are calculated in a similar way:

- **ALGORITHM** CROSS_SECTION_CONTOUR_GENERATOR
- **INPUT:** a rotation angle α and a number of z-values
- **OUTPUT:** A number of contours representing cross-sections through a torus surface at a slope of α degrees
  - ▷ **For each** z-value z_i
    - ▷ **For each** pixel p_{ij} in the plane $C_{z=z_i}$
      - ▷ rotate the coordinates of p_{ij} by $-\alpha$
      - ▷ calculate $d(p_{ij}, T)$
    - ▷ TRACE_TORUS_SURFACE
    - ▷ output contours

```
o ALGORITHM TRACE_TORUS_SURFACE
o INPUT: a 256 × 256 image containing in each pixel its distance to a torus surface
o OUTPUT: One or two closed chains of pixels nearest to the torus surface
    ▷ p_ij = pixel with minimal distance |d|
    ▷ Do
        ▷ Do
            ▷ pij = the adjacent pixel of pij with minimal distance |d|
        ▷ Until chain closed
        ▷ get yet unvisited pixel p_ij with minimal distance |d| < 0.5
    ▷ Until no such pixel found
    ▷ output contours
```

The algorithm TRACE_TORUS_SURFACE simply follows the local minima in the distance image. We call the type of resulting contours *pixel-contours*, that is, each vertex on the contour is a pixel coordinate, and each pair of adjacent vertices has a distance of 1 or $\sqrt{2}$. Besides the fact that such a jagged contour is not very natural—in medical images, displayed objects usually have smooth surfaces—it also increases the amount of data, compared to *vector-contours*. By vector-contours we mean contours whose vertices do not necessarily lie on a pixel grid and where the edge length can take any value. To pass from pixel- to vector-contours, we need a thinning algorithm. The requirements for such an algorithm are:

1. to reduce the number of vertices,

2. to keep the shape visually unchanged.

The second condition is true for a vector-contour whose re-rasterization—for example with the standard Bresenham line algorithm—would give exactly the same pixel-contour at the same magnification. Of course there are an infinite number of such vector-contours. The highest quality should be achieved with spline approximations. We implemented a very simple method: A chain of pixels is replaced by a straight line segment if the greatest orthogonal distance from a pixel to the line segment does not exceed a certain threshold.

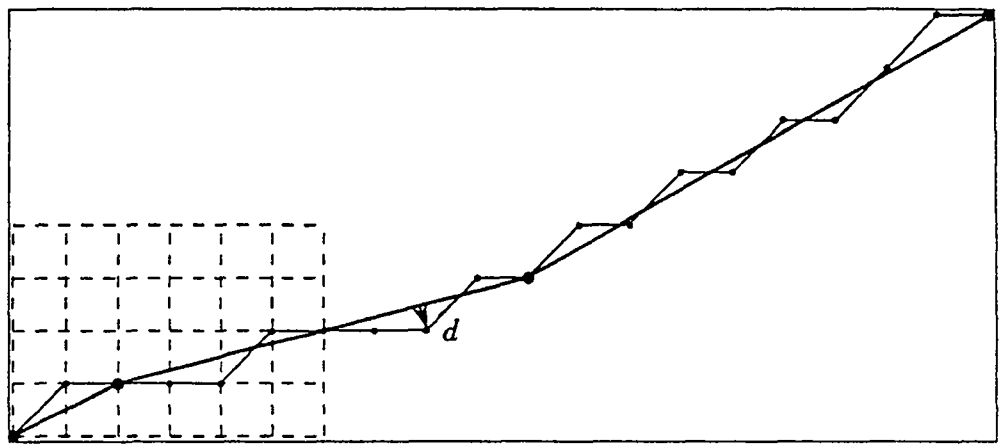


Figure 2.3: Contour approximation with maximal error distance $d < 0.5$ (pixel-contour: 20 vertices, vector-contour (bold): 4 vertices).

- **ALGORITHM** CONTOUR_APPROXIMATION
- **INPUT:** a pixel-contour
- **OUTPUT:** a vector-contour
  - p_{start} = first vertex of pixel-contour
  - **Do**
    - $p_{stop} = p_{start} + 1$
    - continue = TRUE
    - **Do**
      - if max_error($p_{start}, p_{stop} + 1$) < threshold
        - $p_{stop} = p_{stop} + 1$
      - else
        - **if** max_error($p_{start}, p_{stop} + 2$) < threshold
          - $p_{stop} = p_{stop} + 2$
        - **else**
          - continue = FALSE
    - **while** continue
    - add straight line $\overline{p_{start}, p_{stop}}$ to vector-contour
    - $p_{start} = p_{stop}$
  - **while** $p_{start} \neq$ first vertex of pixel-contour
  - output vector-contour

2.4 The test module

We have shown above, that for a torus, $\delta H(\mathcal{P},\mathcal{T}) = \sup_{p\in\mathcal{P}}(\inf_{t\in\mathcal{T}}(d(p,t)))$. Therefore only the distance from a point on $\mathcal{P}$ to $\mathcal{T}$ has to be evaluated. Instead of determining the exact Hausdorff distance, we calculate an approximation on discrete sample points on $\mathcal{P}$. Each surface triangle of $\mathcal{P}$ is covered with points p_i on an orthogonal grid of a distance δ in each direction. We also put points on the triangle edges such that their distance is at most δ. The point with the largest distance from the grid points is situated in the middle of a diagonal segment of length $\sqrt{2}\,\delta$. Since the maximal curvature on a torus is at a radius $\rho = min(R-r, r)$, the maximal error of our Hausdorff distance calculation is

$$\varepsilon = \rho - \sqrt{\rho^2 - \frac{1}{2}\delta^2}.$$

If A is the surface area of $\mathcal{P}$, then the number of sample points N is proportional to A/δ^2. As we mentioned above, both the Hausdorff distance and the difference volume are necessary in order to estimate the shape resemblance. However, calculating the exact intersection of a torus and a polyhedron would require a considerable amount of implementation and computing time. An approximation of the difference volume δV is given by

$$\delta V = \frac{A}{N}\sum_{i=1}^{N}|d_i|$$

where $d_i = \inf_{t\in\mathcal{T}} d(\text{vertex}_i, t)$ and A/N equals the average surface area per sample point. We call d_i the *surface distance error* at vertex i in the sequel.

We also calculate the histograms of the error distribution on the surface of $\mathcal{P}$. We divide the range of sampled surfaces distances into k disjoint adjacent intervals of the same width. In each interval, we plot the number of corresponding sample values divided by the total number of samples. The form of this curve, together with mean value and variance, gives a good apprehension of the quality of the reconstruction.

- **ALGORITHM** COMPARE_POLYHEDRON
- **INPUT:** A polyhedral approximation $\mathcal{P}$ of a Torus $\mathcal{T}$
- **OUTPUT:** δH, δN, $\delta \overline{V}$, statistics
  - ▷ Add points on the surface edges of $\mathcal{P}$ at an interval of δ
  - ▷ Add points on the surface triangles of $\mathcal{P}$ on a grid separated by δ in each direction
  - ▷ Π ={the set of vertices of $\mathcal{P}$} $\cup$ {the set of added points}
  - ▷ **For each** $p \in \Pi$
    - ▷ calculate $d(p, \mathcal{T})$
    - ▷ calculate $\alpha(p, \mathcal{T})$
  - ▷ calculate δH, δN, $\delta \overline{V}$, histograms, mean value, variance

Optionally, we can assume the reconstructed surface triangles on $\mathcal{P}$ no longer to be planar, but to be curved surface tiles. The curvature is determined by the interpolated surface normals. This means that we put the surface points in the place where Gouraud or Phong shading pretends the surface to be. Among the number of parametric surface interpolation methods, we choose the BBG interpolation (Barnhill, Birkhof, Gordon) [31].

At each vertex v_i of $\mathcal{P}$, the normal vector n_i is known. The normal vector defines a tangent plane at v_i. We replace each edge of $\mathcal{P}$ by a parametric cubic curve. In the next step, each triangle of $\mathcal{P}$ is replaced by a BBG interpolation of its edge curves.

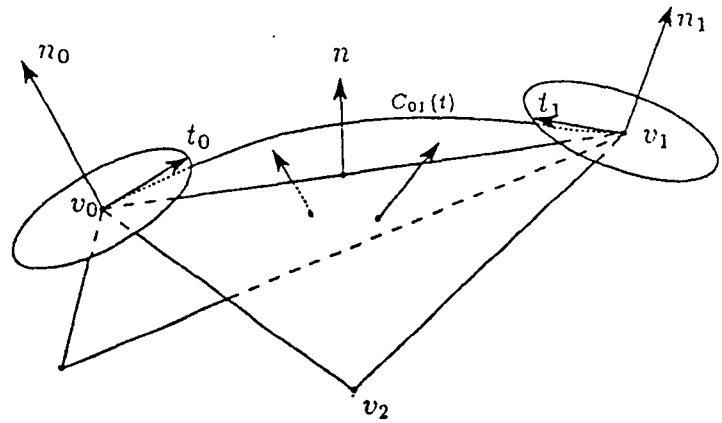


Figure 2.4: Approximation of edges by cubic polynomials.

To obtain the parametric curve of an edge $\overline{v_0 v_1}$ we first calculate the tangent

vectors t_0 and t_1 (see Figure 2.4) given by the following equations:

$$n_0 \cdot t_0 = 0$$

$$t_0 = \mu(v_1 - v_0) + \lambda n$$

$$n_1 \cdot t_1 = 0$$

$$t_1 = \mu(v_1 - v_0) + \lambda n$$

where n is the average normalized normal of the two adjacent triangles sharing the edge. The cubic polynomials that define the curve are of the form

$$C_{01}(t) = at^3 + bt^2 + ct + d \quad 0 \leq t \leq 1$$

The coefficients a, b, c, d are obtained by solving

$$\begin{aligned} C_{01}(0) &= v_0 \\ C_{01}(1) &= v_1 \\ C'_{01}(0) &= t_0 \\ C'_{01}(1) &= t_1 \end{aligned}$$

There is still one remaining degree of freedom: the length of the vectors t_0 and t_1. For our torus surface, we obtained the best results with a length of $0.8 \cdot \|v_1 - v_0\|$.

Given the three polynomials C_{01}, C_{02} and C_{12} of a triangle v_0, v_1, v_2, the BBG interpolation is defined as

$$f(u,v) = \frac{1}{2}(A_1(u,v) + A_2(u,v) + A_3(u,v) - uv_2 - vv_1 - (1-u-v)v_0)$$

where $u, v \in]0,1[, 0 < u + v < 1$ and

$$\begin{aligned} A_1(u,v) &= (1 - \frac{u}{1-v})C_{01}(v) + \frac{u}{1-v}C_{12}(1-v), \\ A_2(u,v) &= (1 - \frac{v}{1-u})C_{02}(u) + \frac{v}{1-u}C_{12}(u), \\ A_3(u,v) &= (1 - \frac{v}{u+v})C_{02}(u+v) + \frac{v}{u+v}C_{01}(u+v). \end{aligned}$$

The function $f(u,v)$ interpolates the three boundary polynomials, the joints between adjacent patches are continuous, but not differentiable (C^0). The algorithm for comparing the interpolated polyhedron is:

- ALGORITHM COMPARE_INTERPOLATED_POLYHEDRON
- INPUT: A polyhedral approximation $\mathcal{P}$ of a Torus $\mathcal{T}$
- OUTPUT: δH, δN, $\delta \overline{V}$, statistics
  - ▷ Calculate the cubic interpolation of the edges of $\mathcal{P}$
  - ▷ Add points on the interpolated curves at an interval of δ
  - ▷ Calculate the BBG interpolation of the surface triangles of $\mathcal{P}$
  - ▷ Add points on the patches on a parametric grid separated by δ in each direction
  - ▷ Π = the set of vertices of $\mathcal{P} \cup$ {the set of added points}
  - ▷ **For each** $p \in \Pi$
    - ▷ calculate $d(p, \mathcal{T})$
    - ▷ calculate $\alpha(p, \mathcal{T})$
  - ▷ calculate δH, δN, $\delta \overline{V}$, histograms, mean value, variance

2.5 Experimental results

We generated cross-sectional images of tori with radii $R = 90$ and $r = 30$, rotated around the x-axis by 0, 45 and 75 degrees (In this section, numbers without explicitly specified units should be read as pixels). The image grid was 256×256. The cross-sections were taken at distances of 2,4,8 and 16. They were placed symmetrically around the center of the tori. In order to see how the error depends on the cross-section placement, we run a series of eight tests, each test consisting of a contour set with fixed slice distance of 4, but having a different z position. From set to set, the cross-section position was shifted in the z direction by 0.5.

From the cross-sections we obtained three different contour types: pixel-contours, vector-contours and hand-drawn contours. The vector-contours were obtained by thinning the pixel-contours with a maximum error of 0.5 pixel. The manually extracted contours were traced with the mouse on the screen. Experience showed that drawing on a magnified image was more comfortable than using the original 256×256 format. We enlarged the images by a factor of 2 in each direction and scaled the contour coordinates back to their original sizes. Additionally, the vector data set was tested using the BBG interpolation. Another set of polyhedra was reconstructed with the marching cube method, using the binary segmented images.

The sample distance on the torus surface was fixed to 0.8, hence the maximal error of Hausdorff measure was $\varepsilon = 5.3 \cdot 10^{-3}$. We calculated the Hausdorff distance δH, the normal difference δN and the difference volume $\delta \overline{V}$. Furthermore,

we used both the sampled surface distance errors and the surface normal errors to calculate histograms, mean values and variances. Additionally, we calculated the volumes and surface areas. The typical results of a contour set is showed in Figure 2.5. The complete list of diagrams for the tori with 75deg slope is included in Appendix B.

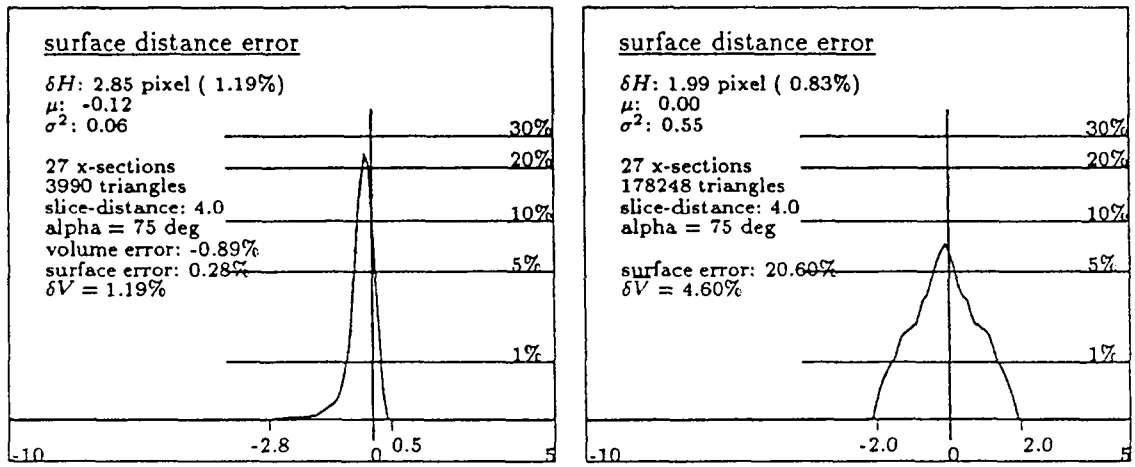


Figure 2.5: Histogram of surface distance error. Left: A polyhedron, reconstructed from vector-contours at a slice distance of 4. The surface distance error lies within the interval [-2.8, 0.5]. About 10% of the sample points have an error of 0. Right: the same set of torus-sections, reconstructed with the marching cube algorithm.

The reconstructed polyhedron was displayed on a graphics workstation with various color codings to visualize the different errors. In this way the worst errors could easily be localized (see Figure 2.6). We made the following observations:

- The pixel-contour edges have a maximum error of 0.5. Since the vector-contours were derived from the pixel-contours, we could expect the error to increase up to 1.0. Surprisingly, the vector-contour edges also stayed within a distance of 0.5. The hand drawn contours however have an error between -1.95 and 1.4 (see Figure 2.7).

- With our method, the number of triangles is twice the number of contour vertices. The data reduction from pixel-contours to vector-contours is 8:1 and from pixel-contours to hand drawn contours 14:1 (see Figure 2.8). For the marching cube example compared to the vector-contours, the ratio of surface triangles ranges from 28:1 to 138:1, depending on the cross-section distance.

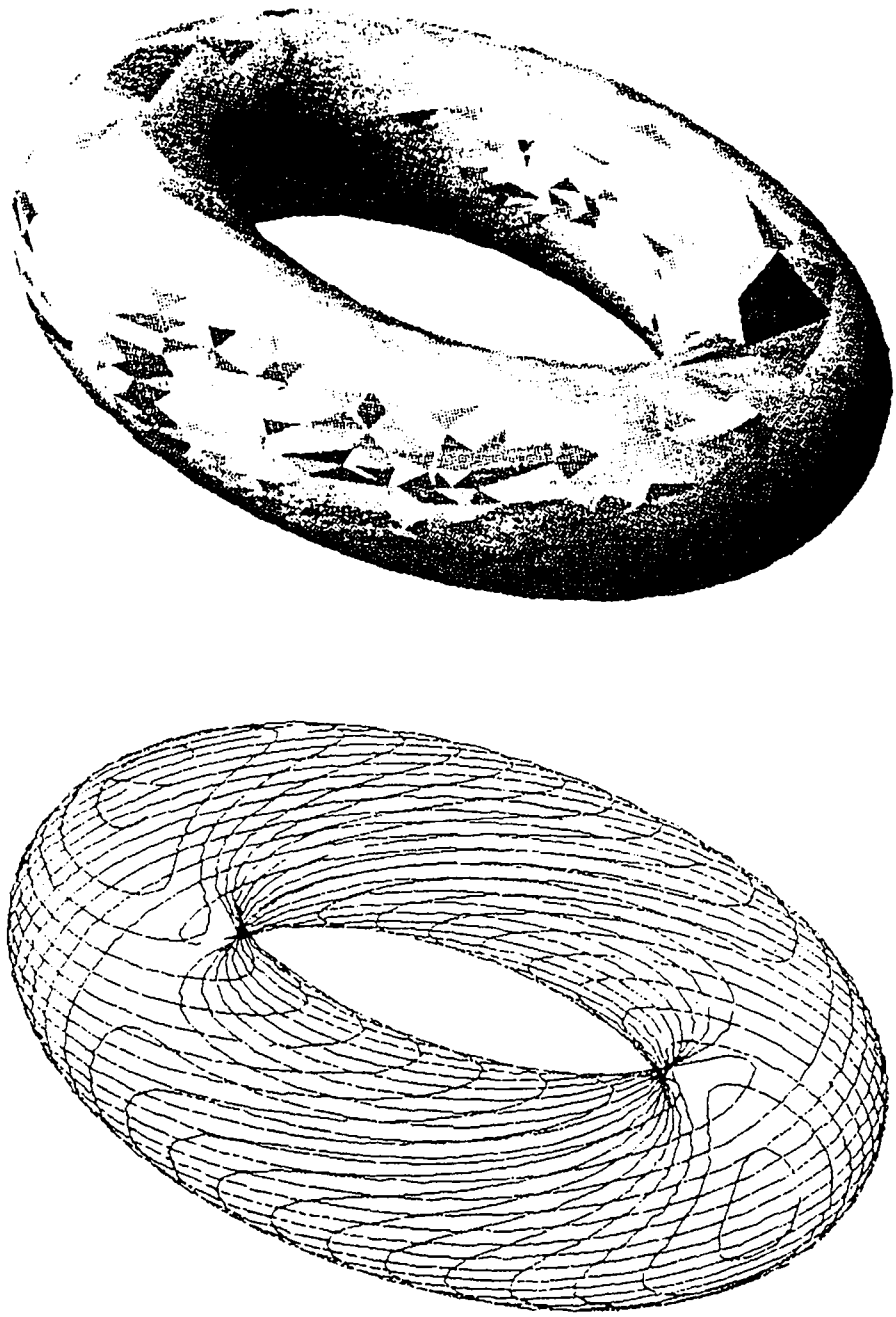


Figure 2.6: The torus cross-sections at a distance of 4 and 75 deg slope (bottom). The reconstruction (top). The color of each triangle depends on its maximum Hausdorff distance. White corresponds to an accurate part, blue corresponds to a negative distance, red indicates an positive distance. The triangles with maximal error (blue) are at places where the torus surface is parallel to the cross-section.

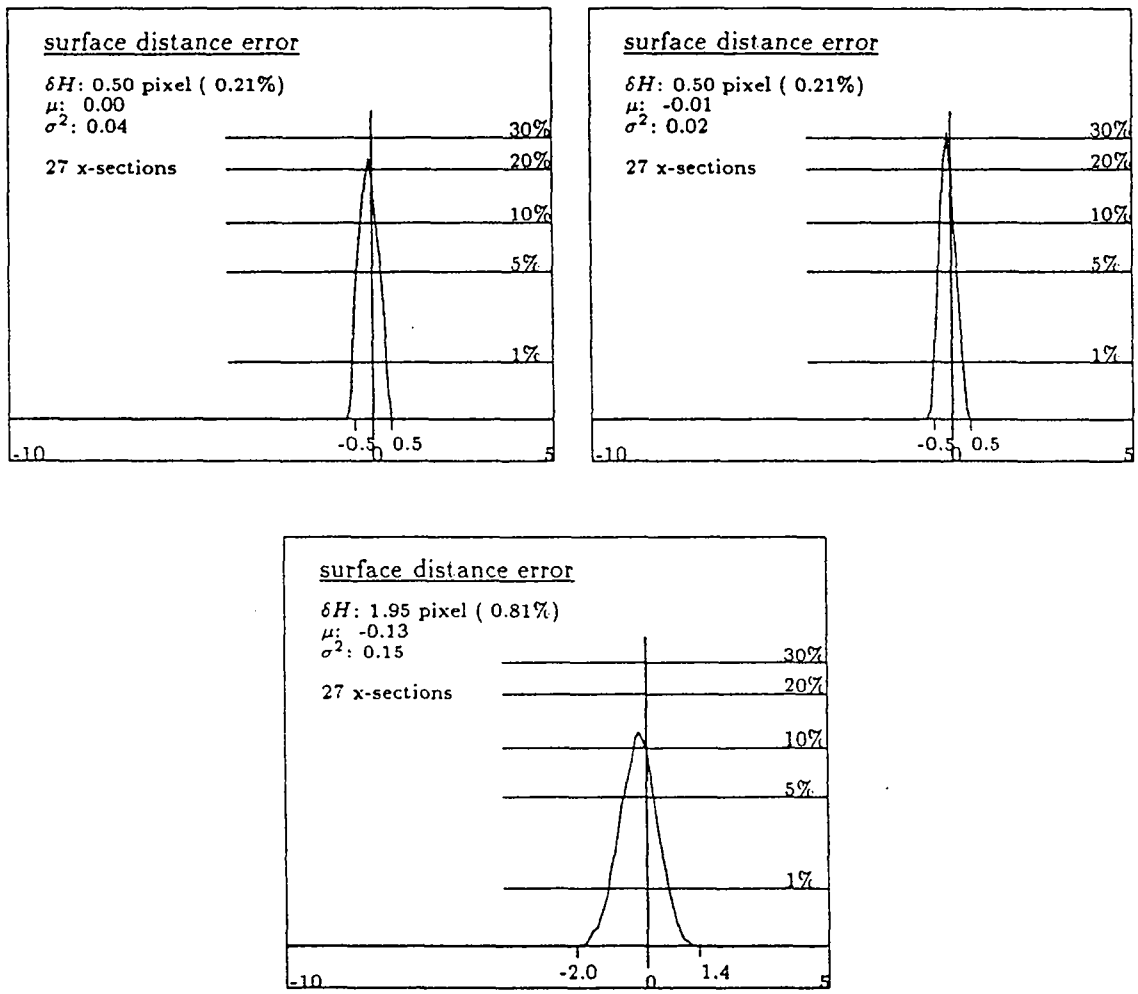


Figure 2.7: Error distribution of the contour approximations. Top left: pixel-contours; top right: vertex-contours, bottom: hand drawn contours.

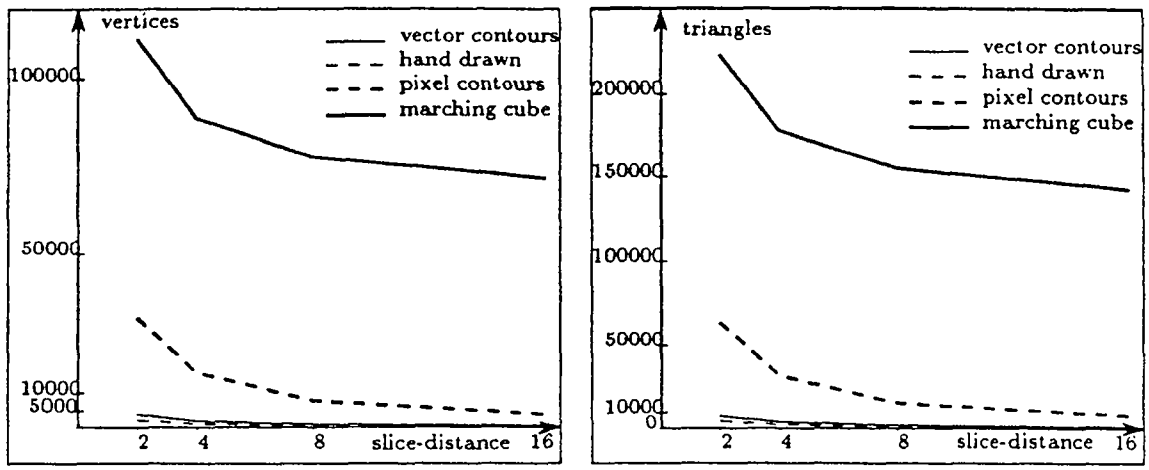


Figure 2.8: Number of vertices (left) and number of triangles (right).

- The worst surface error and normal errors are located at those places where the torus surface is almost parallel to the cross-sections. Consequently, the overall results on the 75 degree example are worse than those of the 45 degree or the upright torus (see Figure 2.9 and 2.10).
- The Hausdorff distance equals the cross-section distance in the worst case. This is due to missing parts at the two extremities. For the marching cube technique, the worst Hausdorff distance is half the slice distance, since it adds intermediate vertices at horizontal faces.
- The sign of δH is generally negative, likewise the mean value. This is due to the fact that the convex part of the torus surface is larger than the concave part. Concave parts are really "saddle-like" parts.
- BBG interpolation leads to a significant reduction of the surface error. The normals however are worse (see Figure 2.9 and 2.11).
- The marching cube reconstructions were worse for nearly all measures. Only the Hausdorff distance was better (due to intermediate cross-sections) and the mean value of the surface error was constantly zero. Especially with increasing distances, the advantage of the Delaunay reconstruction were evident. One reason is the capability of the Delaunay reconstruction to connect contours by triangles spreading over several voxels, whereas the marching cube can only interpolate within one voxel. Another important advantage of the Delaunay reconstruction is the data reduction.

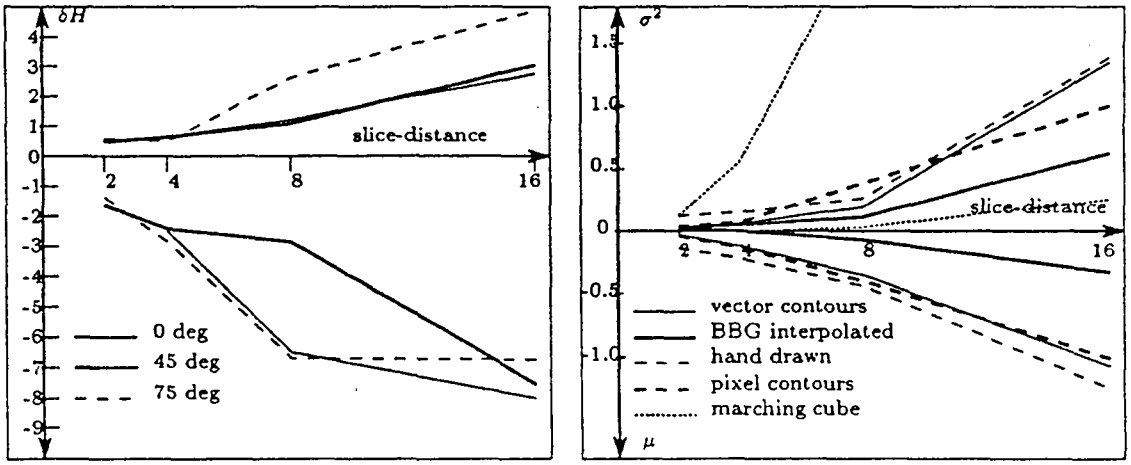


Figure 2.9: The Hausdorff-distance. The surface distance error lies within the two corresponding lines. Left: Vector-contours with different slopes; Right: mean and variance of different contour types at 75 degrees.

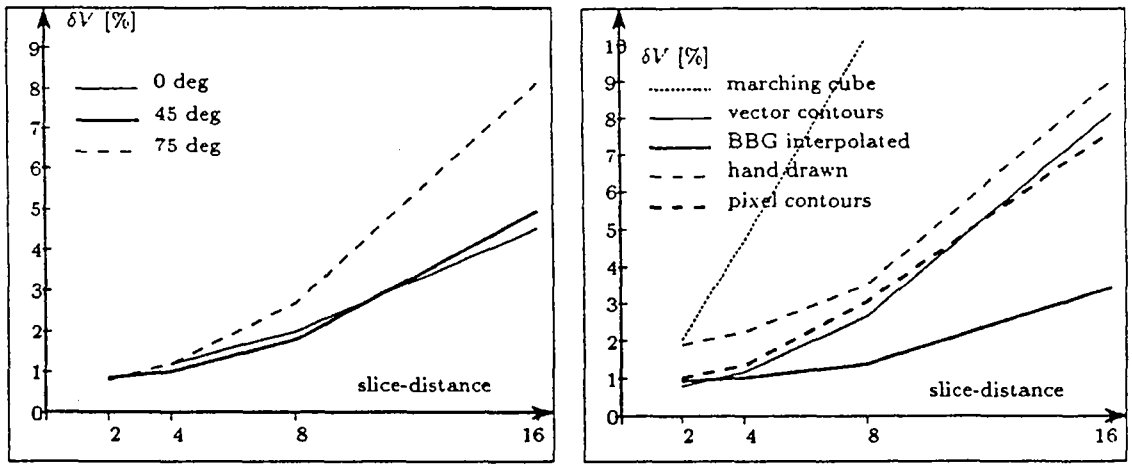


Figure 2.10: Difference volume approximation. Left: Vector-contours with different slopes; Right: Different contour types at 75 degrees.

- The normal error does not decrease with smaller cross-section distances. The best results are between cross-section distances of 4 and 8. When the distance comes near the contour edge error, it is evident that the resulting small triangles may have an arbitrary orientation (see Figure 2.11). We ran a test with a cross-section distance of 0.6 in order to confirm this behavior (see Figure 2.12). Another effect of this oversampling is the increasing surface area. In spite of the high surface distance accuracy, the surface tiles are very irregular, in fact fractal-like (see Figures 2.13 and 2.5).
- Because of similar reasons, the normals of pixel-contours are usually worse than those of vertex-contours. The best results were obtained with hand drawn contours.

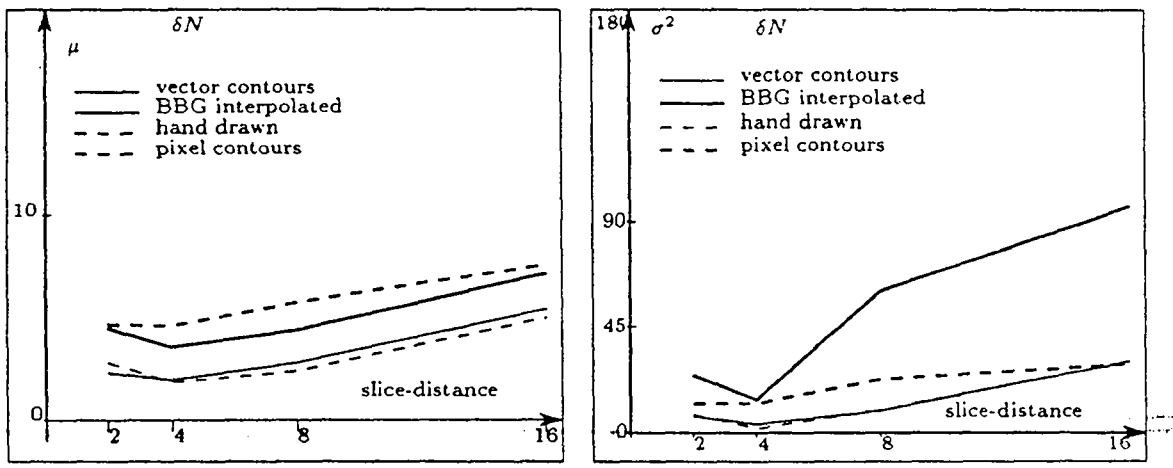


Figure 2.11: Normal error. Left: mean value; Right: variance.

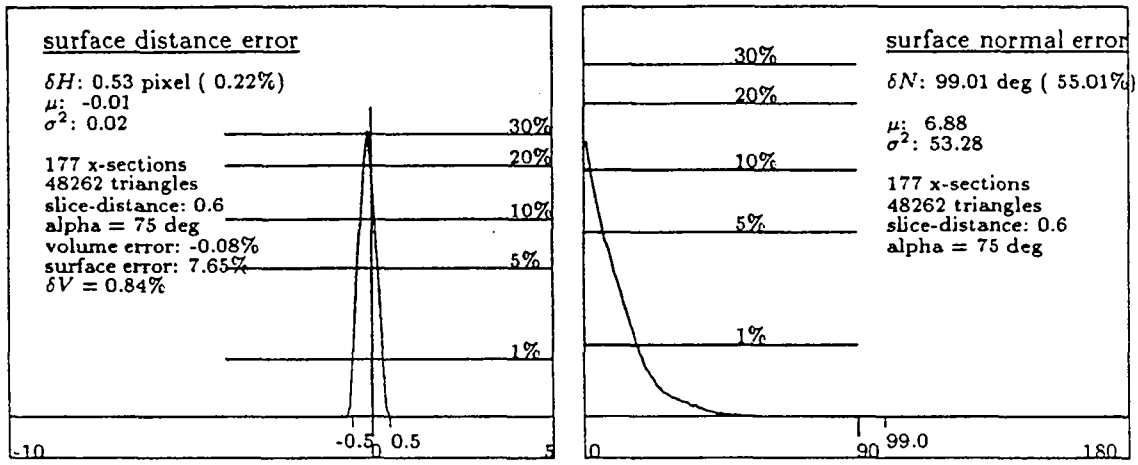


Figure 2.12: A test with a cross-section distance of 0.6. Such an oversampling increases the normal error. Note too the high surface area error of 7.6%.

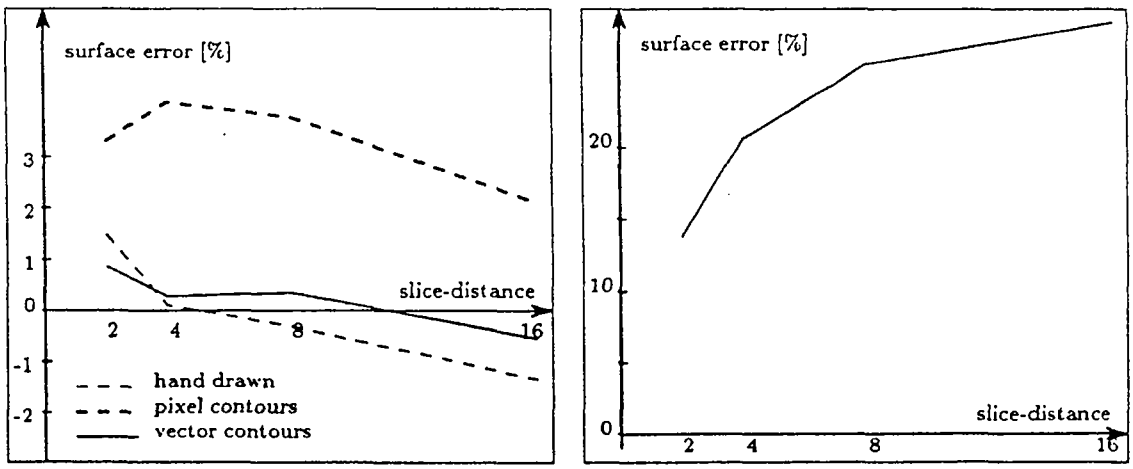


Figure 2.13: The relative surface area error of the 75 degree torus. Right image: marching cube

2.6 Conclusion

We presented a method to measure the quality of 3D polyhedral reconstructions. Our method calculates an approximation of the Hausdorff distance, the difference volume and the surface normal error. Additionally, we calculate the histograms of error distribution.

We tested 3D polyhedra which were obtained by the Delaunay reconstruction presented in the previous chapter, and the marching cube method. The test may be used for any reconstruction method producing polyhedral output.

The test data consisted of artificial cross-sectional images of a torus. Depending on its slope, we get branches and holes. The results may not however be generalized to arbitrary shapes, since a torus shows no sharp edges and is relatively smooth.

One advantage of the Delaunay reconstruction is the considerable data reduction, especially when using a contour thinning method. Such a vertex reduction does not significantly affect the surface distance error, and the quality of the surface normals is actually increased. We found we could further improve the reconstruction quality with the BBG (Barnhill, Birkhof, Gordon) interpolation method. The greatest Hausdorff distance was usually found in Delaunay reconstructions, when compared to marching cube reconstructions. The error distribution and volume differences however were significantly better on the Delaunay models, especially with increasing cross-section distances.

In order to give a better idea of the somehow abstract unit pixel, we shall convert some measures to a real example: Our MR-images provided by a Resonex scanner have a pixel-size of 1.17 mm. The cross-section distances of 2, 4, 8, and 16 correspond to 2.3, 4.7, 9.4 and 18.8 mm distances respectively. One 256×256 image is covering a range of 30 cm. The Hausdorff distance of a torus at a cross-section distance of 4.7mm is then anything up to 3.3 mm, depending on its slope.

Chapter 3

Application: The Simulation of Delivery

3.1 Introduction

An important problem in obstetrics is the prognosis of successful labor. Some of the factors are:

1. the size and shape of the bony pelvis,
2. the size of the fetal head,
3. the force of the uterine contraction,
4. the moldability of the head,
5. the presentation and position of the fetus.

In a small number of cases, a disproportion between the fetal head and the maternal pelvis makes a normal vaginal delivery impossible, or leads to a prolonged course of labor, which may be hazardous for mother and infant. Such a disproportion may be due to a contracted pelvis or due to an enlarged fetal head. The measurements of factors (1),(2) and (5) are possible before the onset of labor. Together with other information like the age of the mother and the knowledge of the outcome of previous labor, they are used to decide whether to do an elective cesarean section at an appointed time or to go into a trial of labor. The trial of labor may result in a successful delivery, or in an emergency cesarean section.

Doing an elective cesarean sections may avoid an otherwise inevitable emergency cesarean section thus preventing mother and child from unnecessary distress. However, not all factors are assessable before the onset of labor. A too cautious prognosis leads to elective cesarean sections done on patients who would

have been able to do a spontaneous delivery. A carefully managed vaginal delivery is usually preferable, even if today the danger of a cesarean section is minimal. Therefore, we have to decrease both the number of emergency cesarean sections and the number of elective cesarean sections by selecting appropriate criteria.

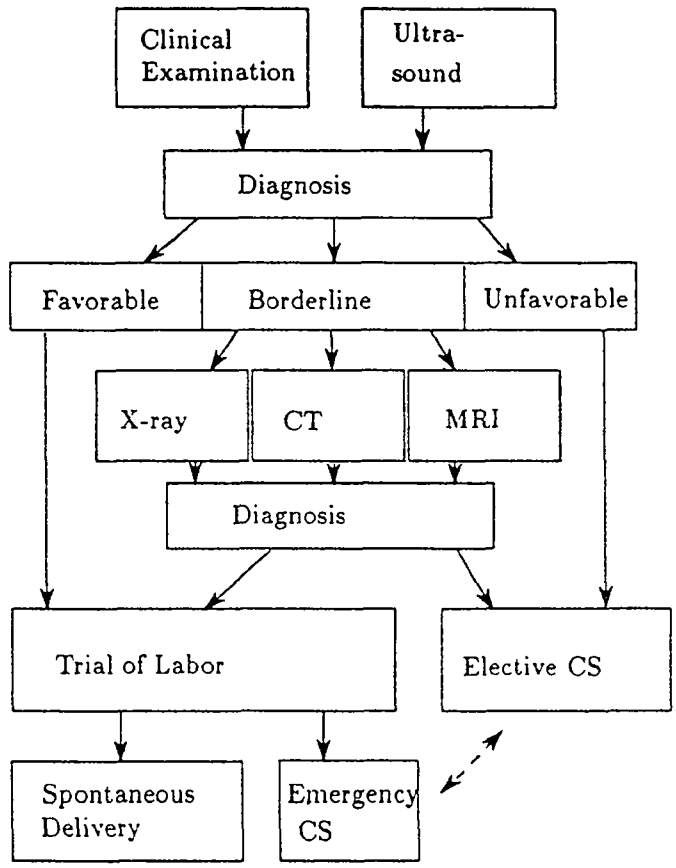


Figure 3.1: The diagnosis of cephalopelvic disproportion.

At the present time, the general procedures to assess the pelvic shape and size are:

- *Clinical measurement:* Some measurements, like the diagonal conjugate and the dimensions of the pelvic outlet can be estimated manually. Usually this examination provides sufficient information for the great majority of cases. For borderline cases however, it may become necessary to get additional and more precise information (The measurement error can be up to 2cm [47]).

- *X-ray pelvimetry:* It provides exact mensuration of the principal diameters. It is also done to get the exact position of the fetus in case of breech presentation. The greatest disadvantage of this method however is the possible hazard of radiation.
- *CT-pelvimetry:* In order to decrease the radiation exposure of the fetus, several authors propose a technique combining digital radiography and CT-tomography ([36, 38]). The technique consists of taking a lateral and an anteroposterior digital radiograph and eventually one axial section to determine the necessary diameters.
- *MRI:* In [37, 42, 46, 50, 52, 53, 54], the authors show that magnetic resonance imaging can be used to accurately measure maternal pelvic dimensions and assess the position of the fetal head. Moreover, this technique has no known hazardous effects on the fetus, as has X-ray exposure. Another advantage is the capability to study soft tissue. Their common belief is that the application of MRI in obstetrics will advance rapidly, and probably replace X-ray based techniques.

The diameters of the fetal head are generally estimated by ultrasound. The main advantages of ultrasound are safety, easy access, low cost and real time capabilities. Unfortunately, this method cannot be applied for the pelvic examination, since bone or bowel gas is obstructing the view. Usually, the deep pelvic structures are not visible ([46, 50]). The articles [39, 45] show that pelvimetry alone is a poor predictor of the outcome of a trial of labor. The need to combine both the head and the pelvic diameters to predict a successful delivery is clearly demonstrated in [40]. The importance of fetal weight is also shown in [49].

In our opinion, the confrontation of the true 3D forms of both pelvis and fetal head would even be more accurate for the prediction of a successful labor, especially since the human pelvis has a very complex shape. Therefore, we have to accomplish two major tasks. Firstly, we must produce 3D models of the pelvis and the fetal head. Secondly, the essential physical mechanisms of delivery have to be formulated and applied to the 3D models. We showed in the first chapter how to create such models based on tomographic images. The 3D models can be computed from magnetic resonance images taken near the end of pregnancy.

The second task now touches a classical domain of robotics: the well known *piano movers problem*, where an object has to be moved from one position to another in the presence of obstacles. The classical way to solve this problem consists in computing an appropriate representation of the set of free configurations (i.e. positions and orientations such that the object does not collide with the environment), the so-called free space (see [34, 48]). These solutions are usually global, that is, they calculate a solution over all possibilities. Additional constraints come from typical robotics applications: angular constraints for robot

arms, limited radius of curvature for cars, and so on. These methods are not easily adaptable for our problem since

- The fetal head and the maternal pelvis are deformable, and we would like our method to take account of this.
- Constraints on position and orientation of the fetal head are different from those encountered in robotics.
- Their complexity depends on the number of vertices, edges or faces of a polygonal robot and environment, and is very high, especially in 3D. Our models consist of several thousand triangles and tetrahedra.

We therefore decided to choose a local method, which is inspired by another problem in robotics: *compliant motion* [41, 44]. Compliant motion occurs in automatic assembly tasks. Fitting parts together generally requires motion between objects in contact. Compliant motion can be produced by a passive mechanical system built in to the manipulator, or by a force control loop implemented in the software (active compliance). We consider the fetal head covered with force sensors and mounted on an imaginary effector. When the head comes in contact with the pelvis surface, we "measure" the forces on the head. The forces are typically along surface normals. The effector produces then small corrective motions to reduce the forces. In this way, we can take into account the physical constraints and even allow a certain interpenetration of the objects to compensate for deformations.

This chapter is organized as follows: our model of forces and constraints applying on the moving body—the fetal head—is explained in the following section. In Section 3.3, we shall describe the algorithm used to find an optimal trajectory. The experimental results are shown in Section 3.4. The last section finally presents a discussion of our method.

3.2 Physical modeling

In the introduction, we invoked the relation between delivery and robotics. We have moving bodies, fixed bodies, forces and moments. However, as we take a closer look onto the details, we recognize that building a valid model of obstetrical mechanics will be a complicated task: the fetus is a highly complex system, with various articulations of different degrees of freedom and mechanical limits. All parts are more or less deformable. The uterine forces are applying from different directions, with different strength and duration. The forces are applying on the trunk which itself is pushing the head. We thus make the following simplifications:

- We only consider the maternal pelvis and the fetal head. In fact, once the head delivered, the shoulders and the rest of the body usually will follow easily. We have to guarantee however that the head cannot execute rotations which would be impossible with the trunk attached.
- The possible positions of the fetus are limited to vertex presentation in our first approach. In particular, we do not allow breech presentation.
- We assume the resultant of the force applying to the center of gravity of the head.
- We do not consider the effects of any kind of soft tissue like uterine musculature or the perineum.

Even with these restrictions, we still have to take into account the deformation of the fetal head and, to a smaller degree, that of the pelvis.

During pregnancy, the mobility of the pelvic joints (symphysis pubis and sacroiliac joints) is increased. This may increase the diameter of the outlet by 1.5 to 2 cm [47], depending on the attitude of the patient. This deformation can be taken into account when measuring the pelvis.

The deformation of the fetal head, also called molding, is accomplished by an overlapping of the bones of the skull at the major sutures. The molding may diminish the biparietal diameter by 0.5 cm without cerebral injury [47]. The degree of molding varies depending on the ossification of the skull, and is not predictable before labor.

Our general idea is to determine a measurement of adequateness by passing the head-model through the pelvis-model. The pelvis is oriented in the following way: The positive x-axis points from the patients right side towards his left side, the positive y-axis is pointing antero-posterior and the positive z-axis is oriented inferior-superior.

We bring the head-model into a defined starting position and orientation above the pelvic inlet. From this point on, we move it step by step downwards in the direction of decreasing z-values. If at any height the head polyhedron penetrates the pelvis polyhedron, we calculate the resulting force and moment applied to the center of gravity g of the head. According to this force and moment, we execute a number of translations and rotations to reduce the force to zero. Persisting forces and moments indicate a position where the head cannot pass without deformation. These forces contribute to a measure of adequateness.

In our model, the fetal head will always pass through the pelvis, even with a severe disproportion. The adequateness value however indicates the quality of the passage. A value of zero corresponds to a head-pelvis configuration, where the head can pass without any deformation required. A positive value indicates the

degree of a necessary molding. The polygonal nature of our models is a reason to allow a moderate interpenetration, since the models present edges and vertices where the real surfaces are smooth. Gliding across such edges may require a small intersection. Also, calculating the complete trajectory even for important disproportions allows us to compare our results to cases where the newborn shows molding. The question where to set the threshold, that is, defining the interval were vaginal delivery would result in too much stress or would be clearly impossible, would require a long term study of borderline cases. Probably the threshold should be chosen in a way to allow a small, reasonable molding.

There are two forces involved in labor: those of the uterus and those of the abdomen, both of which are unknown before the onset of labor. Hence we have no notion of time for the simulation. We therefore divide the z-axis in discrete intervals, and advance the head-model downwards by discrete steps. If the head penetrates the pelvis at a given height, we calculate the resulting force and moment, and execute rotations and translations, yet keeping the head at the same height. Since we fixed the z-position, the head has 5 degrees of freedom: rotation around the 3 axes and translation in the x-y plane. We minimize the force and moment, before advancing to the next z-position.

We define the forces and moments as follows: Let P and H be the polyhedral surfaces of the pelvis and head models. At any point h_i on H we define the force

$$\vec{f}_i = \begin{cases} \vec{0} & \text{if } h_i \text{ outside P} \\ l\,\vec{r}_i & \text{if } h_i \text{ inside P} \end{cases}$$

$$\vec{r}_i = \frac{\vec{n}_i^P - \vec{n}_i^H}{\|\vec{n}_i^P - \vec{n}_i^H\|}$$

where $\vec{n}_i^H$ denotes the interpolated surface normal at point h_i pointing to the outside of H, l is the distance from h_i to P in the opposite direction of $\vec{n}_i^H$ and $\vec{n}_i^P$ is the interpolated surface normal of P at the point $h_i - l\,\vec{n}_i^H =$ P (the point where $h_i - l\,\vec{n}_i^H$ hits the surface of P, see fig 3.2). The force $\vec{f}_i$ thus depends on the surface normals of H and P and on the depth of penetration. Of course this simple model will only be valid if this depth remains relatively small.

The distance l can be calculated efficiently using the tetrahedrization of the pelvis-model. We find the tetrahedron containing h_i. Then we follow the straight line extending from h_i in the opposite direction of $\vec{n}_i^H$. This half line goes through a number of tetrahedra before it hits the surface of P. Furthermore, we may associate a weight to each tetrahedron, and define

$$l_w = \sum_{i=1}^{K} l_i w_i$$

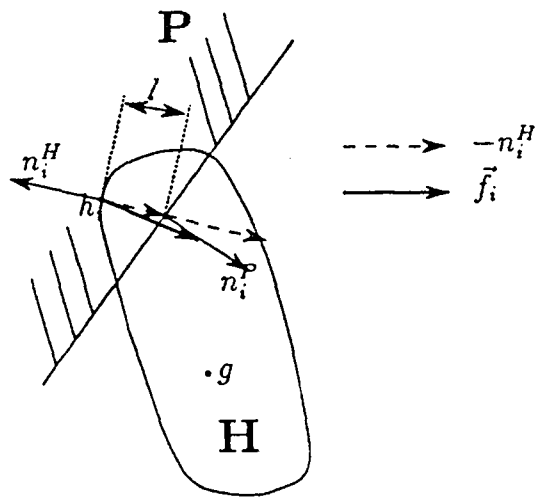


Figure 3.2: The force in point h_i, derived from the normals of the surface of P and H.

where K is the number of tetrahedra we are passing through, l_i is the length of the part of the half line traversing tetrahedron i, and w_i is the weight associated to that tetrahedron. Thus we can locally allow an interpenetration, for example to take into account the mobility of the coccyx (see Figure 3.3). Small values of w_i simulate flexible parts, increased w_i correspond to rigid material.

Some remarks on the complexity of this calculation: in fact the data structure obtained from the Delaunay reconstruction is very efficient for point queries. The first step of the Delaunay reconstruction was to calculate the 3D Delaunay tetrahedrization between each pair of adjacent cross-sections. We got a number of tetrahedra filling the convex hull of the vertices between each pair of cross-section. We call such a tetrahedrization of two adjacent planes a *slice*. In each slice, we deleted in the second step a number of tetrahedra that are not part of the 3D object.

The problem of finding the tetrahedron that contains a given query point can be solved by a search starting in an arbitrary tetrahedron, and advancing from neighbor to neighbor in the direction of the query point, until the containing tetrahedron is reached. The number of visited tetrahedra and hence the query time depends very much on the choice of the starting tetrahedron: the closer it is to the query point, the faster is the search. The tetrahedra set resulting from the Delaunay reconstruction presents two advantages which help to accelerate the search. Firstly, the slices are sorted in increasing or decreasing order, usually along the z-axis. Thus, the z-coordinate of the query point indicates immediately the slice to search in. Moreover, if the query points are given in a specific order—

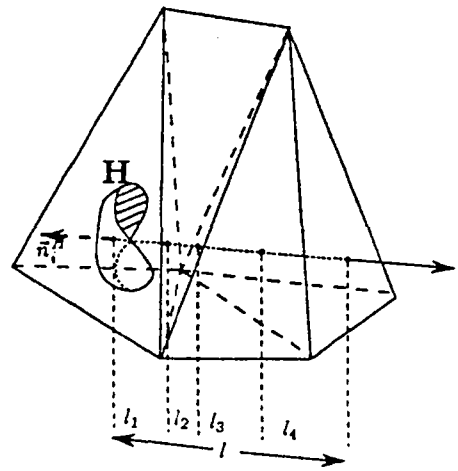


Figure 3.3: Calculation of the force applying at a point of the head.

sorted along contours for example—we can use the tetrahedron found at the previous search as starting tetrahedron. Secondly, we can use the complete set of tetrahedra, those representing the object and those outside the object, that have been marked as deleted. The advantages of using the convex hull of each slice

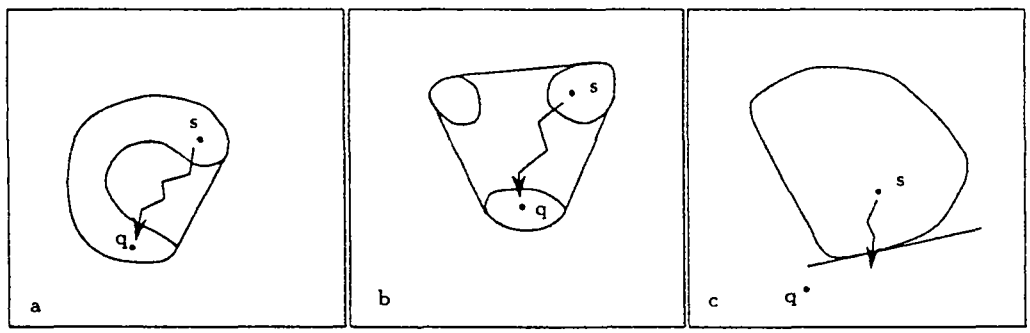


Figure 3.4: Point query in the convex hull of an object slice. From a starting tetrahedra (s), we are searching the tetrahedra circumscribing the query point (q).

are as follows:

- The approach towards the query point will be strictly monotone, even in the case of concave objects (Figure 3.4 (a)).
- In the case of multiple disjoint objects, the choice of a starting tetrahedron inside an object that does not contain the query points does not require

any special consideration (Figure 3.4 (b)).

- If the query point lies completely outside the tetrahedra set, we can stop the query once we leave the tetrahedra set. This follows from the fact that each slice is convex (Figure 3.4 (c)).

If n_t is the number of tetrahedra of the pelvis, and n_s is the number of slices, and we suppose an average number of n_t/n_s tetrahedra per slice, the query for one point takes $O(n_t/n_s)$ time in the worst case. However, in cases of a sorted distribution of points to search, an appropriate choice of the starting tetrahedron yields a constant time query.

We calculate the forces on the head at N discrete sample points, distributed on the surface of the head. These sample points are obtained by adding vertices inside each surface triangle on a regular grid with adjustable dimensions. Each sample point thus represents an average area of a/N, if a is the surface of the head. The resulting translation force $\vec{t}$ on a point g inside the head is the sum of the forces $\vec{f_i}$:

$$\vec{t} = \frac{a}{N} \sum_{i=1}^{N} \vec{f_i}.$$

The resulting moment $\vec{m}$ is the sum over all i of the component of $\vec{f_i}$ orthogonal to $(g - h_i)$:

$$\vec{m} = \frac{a}{N} \sum_{i=1}^{N} \vec{f_i} \times (g - h_i).$$

Now we have to map the translation force $\vec{t}$ to a translation $\vec{tr}$ and the moment $\vec{m}$ to a rotation of θ degrees around the vector $\vec{m}$. The translation takes place in the direction of $\vec{t}$. We make its length linearly dependent on the length of $\vec{t}$, with an upper bound of tr_{max}:

$$\vec{tr} = \begin{cases} \mu \vec{t} & \text{if} \quad \mu \|\vec{t}\| < tr_{max} \\ tr_{max} \frac{\vec{t}}{\|\vec{t}\|} & \text{else} \end{cases}$$

The translation is described by the 4×4-matrix

$$\mathcal{T}(\vec{tr}) = \begin{bmatrix} 1 & 0 & 0 & tr_x \\ 0 & 1 & 0 & tr_y \\ 0 & 0 & 1 & tr_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The maximal rotation θ around $\vec{m}$ is limited to θ_{max}. We define it as

$$\theta = \begin{cases} \lambda \|\vec{m}\| & \text{if} \quad \lambda \|\vec{m}\| < \theta_{max} \\ \theta_{max} & \text{else} \end{cases}$$

To execute the rotation around $\vec{m}$, we make a coordinate system transformation. After normalization of $\vec{m}$, the transformation matrix is given by

$$\mathcal{A} = \begin{bmatrix} \frac{m_y}{\sqrt{m_x^2+m_y^2}} & \frac{m_x m_z}{\sqrt{m_x^2+m_y^2}} & m_x & 0 \\ -\frac{m_x}{\sqrt{m_x^2+m_y^2}} & \frac{m_y m_z}{\sqrt{m_x^2+m_y^2}} & m_y & 0 \\ 0 & -\sqrt{m_x^2+m_y^2} & m_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

The transformed basis is an orthonormal basis, with the z-axis in the direction of $\vec{m}$. In the transformed coordinate system, the rotation matrix is

$$\mathcal{R}_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

The rotation matrix $\mathcal{M}(\theta)$ is then given by

$$\mathcal{M}(\theta) = \mathcal{A}\,\mathcal{R}_z(\theta)\,\mathcal{A}^{-1}.$$

We describe the head position relative to its original position. Each time we update position and orientation of the head, we move it into the origin, apply rotations and move it to the final position. The position and orientation of the head at a discrete step i are thus determined by a matrix $\mathcal{S}_i$ composed of the translation matrix $\mathcal{T}(-g)$, which moves the head into the origin, a rotation matrix $\mathcal{R}_i$ and a translation matrix $\mathcal{T}_i$ which defines the head position relative to the origin:

$$\mathcal{S}_i = \mathcal{T}_i \mathcal{R}_i \mathcal{T}(-g)$$

After calculation of $\vec{tr}$, θ and $\mathcal{M}$ for $\mathcal{S}_i$, we get a new position

$$\mathcal{S}_{i+1} = \mathcal{T}_{i+1} \mathcal{R}_{i+1} \mathcal{T}(-g),$$

where

$$\mathcal{R}_{i+1} = \mathcal{M}(\theta)\mathcal{R}_i$$

and

$$\mathcal{T}_{i+1} = \mathcal{T}(\vec{tr})\mathcal{T}_i.$$

Finally, we define the adequateness function ε_k as

$$\varepsilon_k = \frac{a}{N} \sum_{i=1}^{N} \|\vec{f}_i\|^2$$

with the head in position $\mathcal{S}_k$.

3.3 Simulation

For the simulation of delivery, we bring the head in a starting position and orientation $\mathcal{S}_0$. This can be the original head position and orientation given by the MR-Image. We can also interactively select other positions and orientations. We take the center of gravity of the head as reference point g, that is the point in which we calculate the resulting force and moment. In the optimization step, we fix the z-coordinate of this point. This means that the head can move in the x-y plane, and rotate around g.

Our goal is to reduce the force applied on the head to zero or to the smallest value possible with $\mathcal{S}_i\, g$ constrained in the plane $z = z_i$. Therefore, we calculate θ, $\mathcal{M}$ and $\vec{tr}$ —we set tr_z to zero, since the z-position is fixed—and bring the head into a new position and orientation. If necessary, we repeat these steps, until we find a position $\mathcal{S}_{opt}$ where the forces are zero or at a minimum. Once the optimal position $\mathcal{S}_{opt}$ is found, we calculate the adequateness value ε_{opt}. Then we move the head downwards by a value of Δz and get $\mathcal{S}_1$. We repeat the optimization procedure. We stop the simulation if the head reaches a z-coordinate z_{end} sufficiently below the pelvis. For each z-coordinate z_i $0 \leq i < (z_{end} - z_0)/\Delta z$, we thus get the adequateness value ε_i indicating the persistent interpenetrations. Consequently, if not all ε_i are equal or close to zero, we claim that the head cannot pass without deformations.

- **ALGORITHM** SIMULATION (H, P,$\mathcal{S}$)
- **INPUT:** Head model **H**, pelvis model **P**, start position of **H**
- **OUTPUT:** Head trajectory, adequateness function ε
  - Set $\mathcal{S}_0$ to start position of **H**
  - $z = z_0$
  - $i = 0$
  - **Do**
    - $\mathcal{S}_{opt}$ = OPTIMIZE($\mathcal{S}_i$)
    - ε_i = adequateness value of $\mathcal{S}_{opt}$
    - **Output** ($\mathcal{S}_{opt}, \varepsilon_i$)
    - Set $\vec{tr} = (0, 0, -\Delta z)$
    - $\mathcal{S}_{i+1} = \mathcal{T}(\vec{tr})\mathcal{S}_{opt}$
    - $z = z - \Delta z$
    - $i = i + 1$
  - **While** ($z > z_{end}$)

The optimization step consists in repeatedly calculating the translation force and moment in g, and updating the position of the head. If the moment and force decrease, we note the new position as optimum. However, executing the indicated rotations and translations may not always decrease the parameters. Some of the reasons are:

- The executed movements were too big, because we chose the variables θ_{max} and tr_{max} too large. In this case, the head will typically oscillate.
- Since we calculate the forces on discrete sample points, appearance of a new point leads to a discontinuity in the forces.
- Due to our simplified model, there are configurations where the execution of the calculated forces can push the moving body further into the obstacle. We call these positions balanced, but not stable.

We apply a simple procedure to cover these cases:

- If $\vec{m}$ is oscillating, we decrease θ_{max}.
- If $\vec{tr}$ is oscillating, we decrease tr_{max}.
- If $\vec{m}$ is slowly decreasing or constant, we increase θ_{max}.
- If $\vec{tr}$ is slowly decreasing or constant, we increase tr_{max}.

Each time the moment and the force increase, we augment a counter. If the counter reaches a maximal value, we stop the optimization and return to the optimal position. We use the sum of moment and force as optimization parameter. Minimizing only the variable ε does not affect the orientation of the head, since the definition of ε does not contain the moment. Minimizing the moment leads to an optimal adaptation to the pelvic geometry. If the sum of moment and force are optimal, ε is most likely to be optimal, too.

- **ALGORITHM** OPTIMIZE($\mathcal{S}_0$)
- **INPUT:** Head position $\mathcal{S}_0$
- **OUTPUT:** Optimal head position $\mathcal{S}_{opt}$
  - ▷ Set $\mathcal{O} = \mathcal{S}_0$
  - ▷ Set $\xi_{opt} = \|\vec{m}\| + \|\vec{t}\|$
  - ▷ **if** $\xi_{opt} = 0$ **return**($\mathcal{O}$)
  - ▷ $k = 0$
  - ▷ counter = 0
  - ▷ **Do**
    - ▷ Calculate $\theta, \mathcal{M}(\theta), \vec{tr}$ for $\mathcal{S}_k$
    - ▷ **if** H is oscillating or constant
      - ▷ Change θ_{max} and tr_{max}
    - ▷ Set $tr_x = 0$
    - ▷ $\mathcal{S}_{k+1} = \mathcal{T}(\vec{tr})\,\mathcal{T}_i\,\mathcal{M}(\theta)\,\mathcal{R}_i\,\mathcal{T}(-g)$
    - ▷ $\xi = \|\vec{m}\| + \|\vec{t}\|$
    - ▷ **if** $\xi > \xi_{opt}$
      - ▷ worse
      - ▷ counter = counter + 1
      - ▷ **if** (counter > MAX) **return**($\mathcal{O}$)
    - ▷ **else**
      - ▷ $\xi_{opt} = \xi$
      - ▷ $\mathcal{O} = \mathcal{S}_{k+1}$
    - ▷ $k = k + 1$
  - ▷ **While** ($\xi > 0$)
  - ▷ **Return** ($\mathcal{O}$)

3.4 Experimental results

The implementation of the simulation program was done in C. The program was divided into three independent modules, the force calculation and simulation module, a visualization module and a server acting as a first-in-first-out buffer. The calculation module is relatively machine independent and is intended to run

on a fast machine. It communicates the results—the 4 × 4 Matrix describing the position and orientation of the head and the adequateness value—via RPC (remote procedure call) to the server. The display module runs uniquely on a Silicon Graphics computer. Its task is to fetch the results from the FIFO server and to display head and pelvis. In this way, the simulation is accelerated, because calculation and display are done in parallel. Furthermore the application can be run on distributed networks. On the display module, we have additional functions: We can visualize the locus and strength of the forces on the head by changing the colors of intersecting vertices. We can interactively measure distances on the 3D models, and we may pick specific triangles and change their color. This may be useful to mark the small and large fontanels on the model, which helps to identify the orientation of the fetal head during the simulation.

A major problem was to find suitable MR images taken during pregnancy. Therefore we used pelvic images of a nonpregnant person. The pelvis was reconstructed from 23 axial MR images, separated by a distance of 0.86cm. We measured the following diameters on the pelvis-model:

		model [cm]	normal* [cm]
pelvic inlet	obst. conjugate	12.0	> 10.0
	transverse	13.4	13.5
	transverse median	13.4	13.0
	oblique	13.2	13.0
midpelvis	interspinous	10.8	10.0
	anterior-posterior	12.3	> 11.5
pelvic outlet	anterior-posterior	11.8	9.5-12.0
	transverse	12.2	11.0

(*) As described in [47].

The criterion of Magnin (*Indice de Magnin*, see [43]) is defined as the sum of transverse median diameter and obstetric conjugate, and is 25.4 cm in our example. That corresponds to a favorable measure (The prognosis is favorable if the sum is $\geq$ 23 cm, unfavorable if the sum is $<$ 20 cm [43]). This criterion does not take into account the head diameters.

The head was reconstructed from 16 MR scans of an adult. We changed manually the proportions to obtain a resemblance with a newborn. The diameters were then scaled to the sizes given in [43]. The biparietal diameter was 9.4 cm in this model (average size). Since the pelvis was not contracted, we used a second model having the same proportions, but scaled to a biparietal diameter of 11cm to simulate a disproportion. The head was covered with additional vertices to

reach a given maximal distance between vertices. Optionally, we used the BBG interpolation method described in the previous chapter to calculate the additional vertices. Figure 3.5 shows two views of the head.

For the simulation step, we restricted the starting position of the head to one of the vertex presentations: right (R) or left (L); occiput (O); posterior (P) or transverse (T) or anterior (A) position.

We executed a large number of runs with different parameters. The goal was to verify the robustness, but also to reduce the number of calculation steps and thus to accelerate the execution time. The computation of a complete trajectory as presented in Figure 3.6 took about one hour of CPU time on a DECstation 5000. In Figures 3.7 to 3.9 we compare the different trajectories obtained with different parameters and different presentations of the fetal head. In each figure, we show the adequateness functions of two trajectories and the maximum angular distance between the corresponding head orientations at each z-coordinate. The abscissa represents the z-coordinate.

Figure 3.6 shows the trajectory of the large head-model starting from ROP (right occiput posterior) presentation. Although our model is limited to the bony structures, the head makes exactly the same movements as illustrated in [47]. It enters the pelvic inlet with its sagittal suture directed transversely, then performs an internal rotation that brings the occiput towards the symphysis pubis. The fact that the head follows the principal movements as described in the literature validates our physical model. Below each image, the adequateness function is plotted; a vertical line indicates the current head position. Apparently, the passage of the outlet is somewhat constrained.

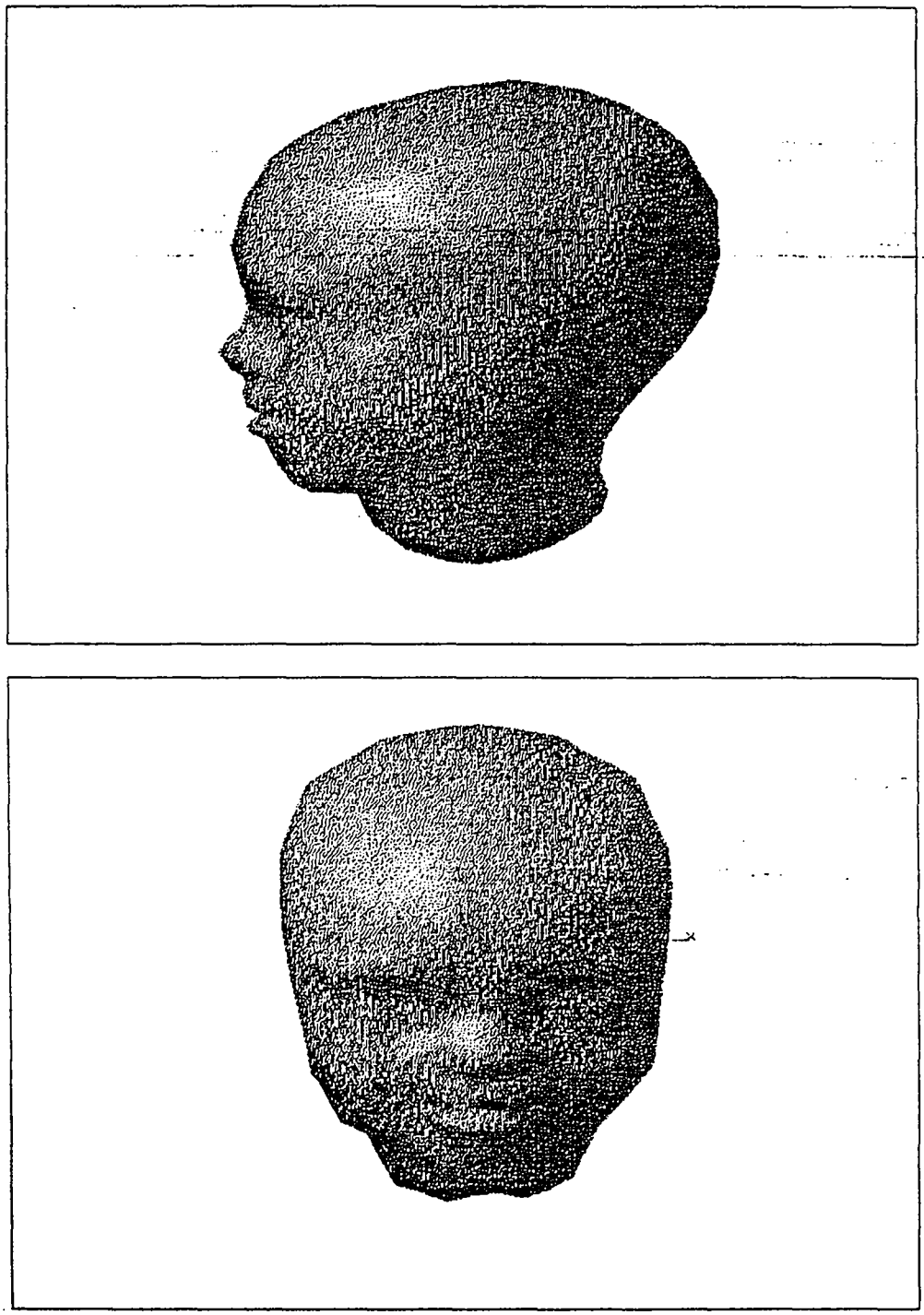


Figure 3.5: The fetal head reconstructed from 16 MR scans, with contour modifications.

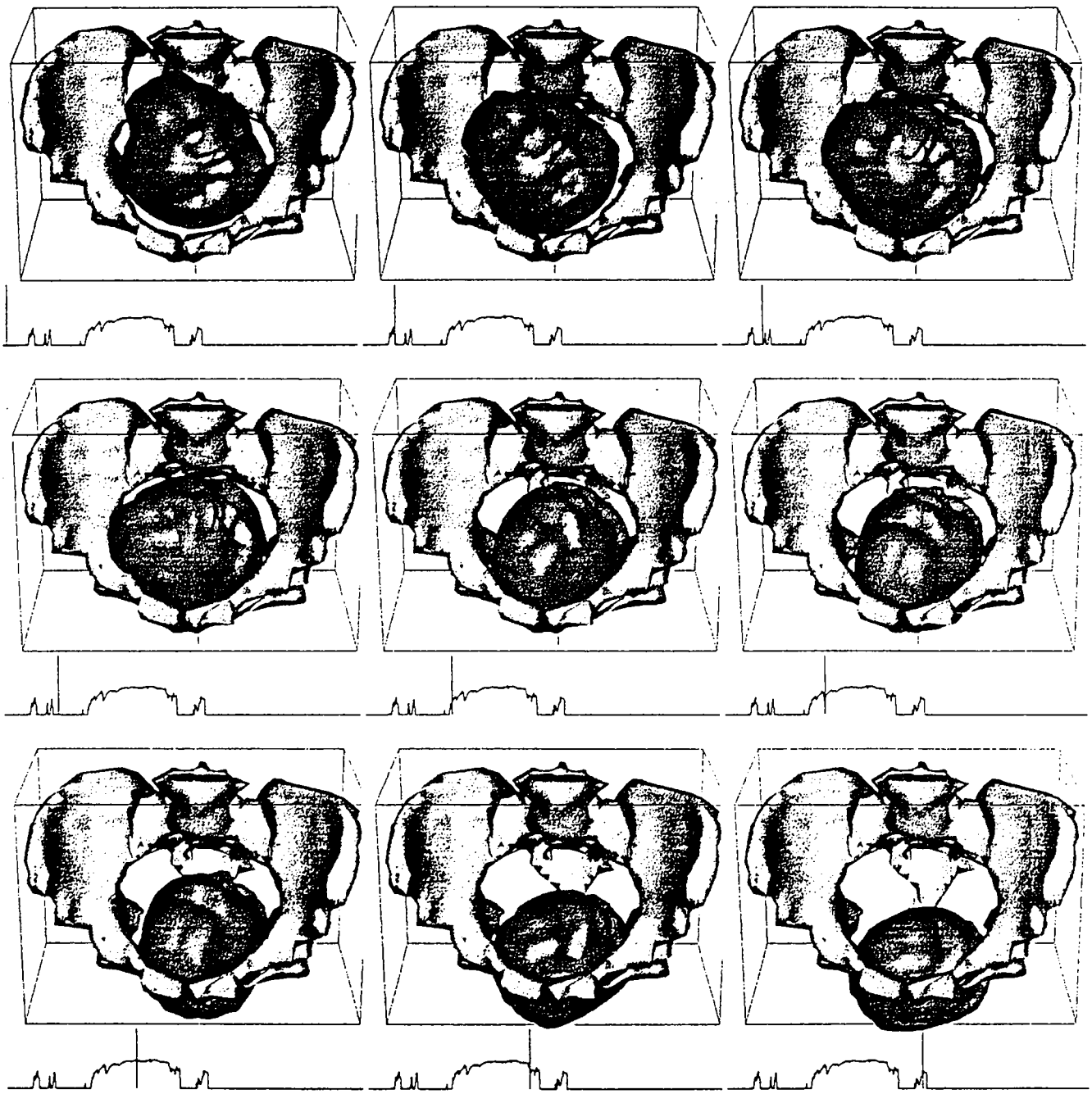


Figure 3.6: The fetal head traversing the pelvis from ROP presentation: descent, internal rotation and extension.

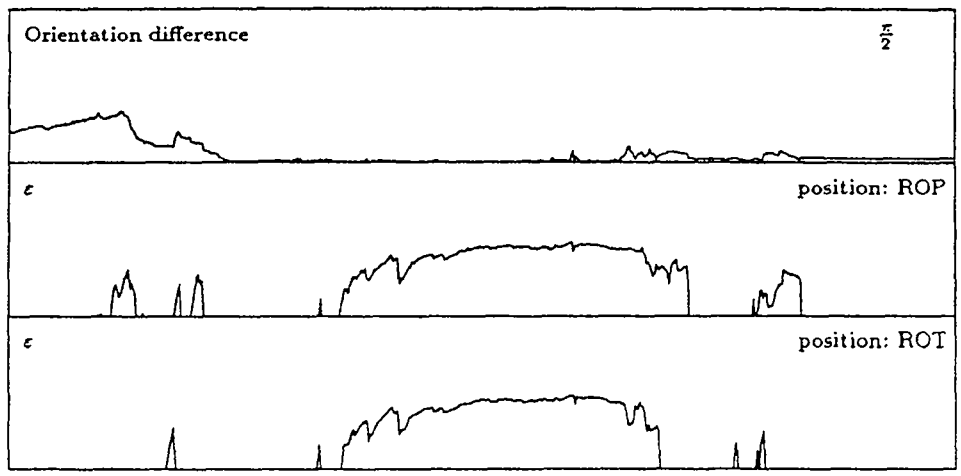


Figure 3.7: Comparing Right Occipito Transverse (ROT) and Right Occipito Posterior (ROP) presentation. After the difference in the beginning, the two trajectories nearly coincide.

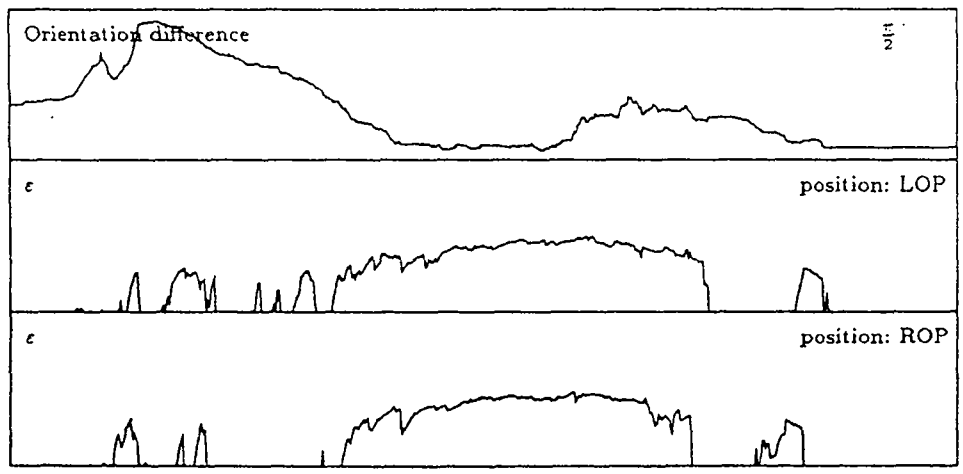


Figure 3.8: Comparing Right Occipito Posterior (ROP) and Left Occipito Posterior (LOP) presentation. The adequateness functions are similar.

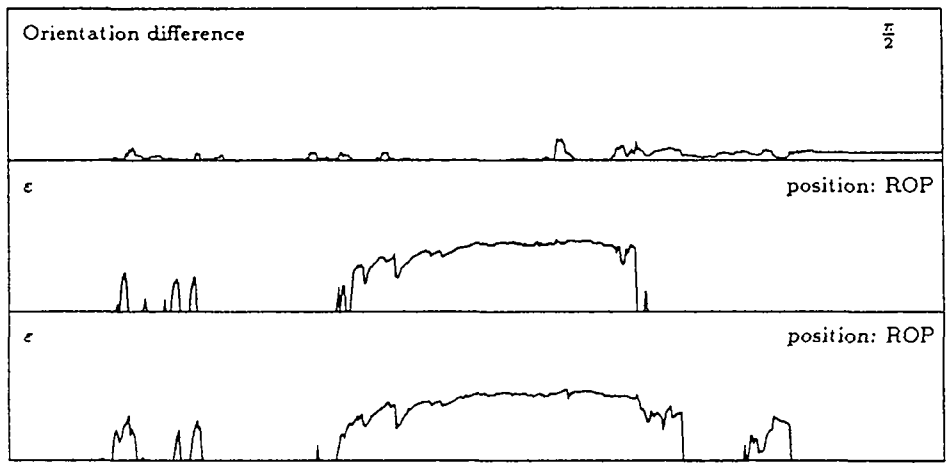


Figure 3.9: Right Occipito Posterior (ROP) presentation with different simulation parameters. The number of retries in the case of increasing optimization parameter was set to 3 (bottom) and 10 (center). The CPU time was 69 and 148 min. resp. while the adequateness function was not too different.

3.5 Discussion

We are able to provide good 3D pelvic models from MRI, even with only a few cross-sections. The accuracy of our reconstruction method is shown on algebraic surfaces in Chapter 2. However, these accuracy results may not be generalized to pelves. The accuracy should be verified in the future; for example with images of cadaver pelves.

All diameters of the pelvis can be evaluated interactively on the models. In this way, we obtain more information than with X-Ray pelvimetry. Moreover, the volume of the fetal head can be approximated, which would be useful in studying growth.

We are able to model the physical mechanism of delivery, restricted to the bony pelvis and fetal head. The 3D confrontation of the fetal head and the pelvis gives indications on the success of labor, which should be more accurate than comparing a number of diameters. As for most simulations, it is difficult to determine whether the model is valid and if conclusions can be derived from the simulation results. Up to now, the encouraging features have been the realistic movement of the head in all the tests (which corresponds to the trajectory described in literature), and the repeatability with different parameters. We have just started to explore the potential of 3D modeling and motion planning. The major part of work must still be done:

- The fetal head has been produced artificially. It may not correspond to a head of a newborn. It has to be shown that a real fetal head will also be assessable from MR images using our method.
- We worked with one single model of pelvis and head. It is difficult to generalize such results. We have to verify the results on a large number of different cases, especially with pelves from different pelvic types.
- We have to show that the adequateness function is statistically significant for a prediction of a successful labor.

Of course a long term collaboration with a department of obstetrics would be necessary to establish statistics and to evaluate the possible benefit of this method. The first stage would be a kind of supervised learning by studying the outcome of borderline cases which went into a trial of labour. For this purpose, the necessary measurements may also be taken after the delivery. The goal of this step is to estimate the different threshold values of the adequateness functions for borderline cases. In a second stage, the method may be used parallel to conventional diagnosis, but without influencing the decisions. If this step shows a sufficient statistical significance, a future use of this method may be appropriate.

Figure 3.10: Un accouchement dans une riche famille parisienne. D'après une gravure d'Abraham Bosse (1602–1676) Paris, Bibliothèque Nationale.

Appendix A

On connecting two simple polygons

THEOREM. *Given two parallel planes* Π_1 *and* Π_2 *and two simple closed polygons* $P_1 \in \Pi_1, P_2 \in \Pi_2$, *it is always possible—by adding at most 2 Steiner points—to connect the vertices of* P_1 *to the vertices of* P_2 *in order to obtain a simple closed polyhedron whose intersection with* Π_i *is* $P_i (i = 1, 2)$.

DEFINITIONS. A *polygonal arc* joining two points a and b in $\mathbb{R}^N$ is a sequence of straight segments placed end-to-end, starting at a and finishing at b. A polygonal arc is called *simple* if it has no points of self-intersection. A *simple closed polygon* is a simple polygonal arc with $a = b$.

If S is a collection of triangles, then S satisfies the *intersection condition* if two triangles of S either are disjoint or have exactly one vertex in common or have two vertices, and consequently the entire edge joining them, in common.

S is *connected* if it satisfies the intersection condition and there is a path along the edges of the triangles from any vertex to any other vertex.

If we consider a vertex v of a collection S of triangles satisfying the intersection condition, then the edges opposite v in the triangles having v as a vertex will form a graph, called the *link of* v.

A *simple closed polyhedron* is a collection S of triangles such that

1. S satisfies the intersection condition.

2. S is connected

3. for every vertex v of a triangle of S, the link of v is a simple closed polygon.

PROOF. Let P and Q be two simple polygons in the planes $z = z_p$ and $z = z_q$

($z_p > z_q$). In each plane the vertices are assumed to be in general position, i.e. there are no 3 successive vertices on a straight line.

We begin by connecting each vertex of P with the leftmost[1] vertex l of Q and each vertex of Q with the rightmost vertex r of P, thereby obtaining two non-intersecting cones having the edge $\overline{rl}$ in common. The cone (P, l) lies above the plane parallel to the y-axis and passing through l and r. The cone (Q, r) lies below that plane. Let $\wp$ be the (not simple) closed polyhedron cone$(P, l) \cup$ cone(Q, r).

Now we consider the edges of P and Q incident to r and l (see Figure A.1). We project the edges $\overline{q_1 l}$ and $\overline{l q_2}$ along the line $\overline{lr}$ onto the plane $z = z_p$, obtaining $\overline{q_1' r}$ and $\overline{r q_2'}$. We name the angles $\alpha = \angle p_1 r p_2$); $\beta = \angle q_1' r q_2' = \angle q_1 l q_2$; $\gamma = \angle q_2' r p_1$

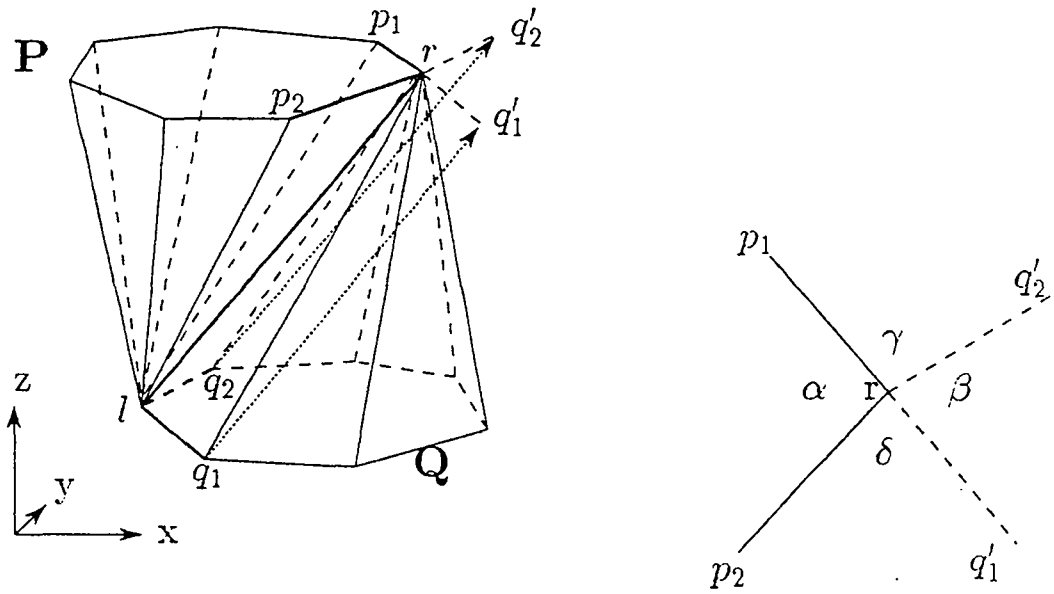


Figure A.1

and $\delta = \angle p_2 r q_1'$. Obviously $\alpha + \beta + \gamma + \delta = 2\pi$. Since l and r are leftmost and rightmost points and P and Q are simple, we have $0 < \alpha < \pi$, $0 < \beta < \pi$, $\gamma > 0$ and $\delta > 0$. So at least one of the angles γ and δ has to be smaller than π. Otherwise $\alpha + \beta = 2\pi - \gamma - \delta < 0$.

Without loss of generality we assume that $\delta < \pi$. We call the area defined by $\overline{rp_2}$ and the two half-lines starting in r and p_2 and being parallel to $\overline{lq_1}$ the *shadow of the edge* $\overline{rp_2}$, $S(r, p_2)$ (see Figure A.2). Likewise, $S(l, q_1)$ is the area defined by $\overline{lq_1}$ and the two half-lines starting in l and q_1 and being parallel to

[1]We mean by leftmost (rightmost) the vertex with the smallest (greatest) x-coordinate. In case of multiple such points we rotate our coordinate-system around the z-axis until we find a unique leftmost (rightmost) point.

$\overline{rp_2}$. If $S(r, p_2)$ contains any other vertex of P, we add a Steiner point p_s on the

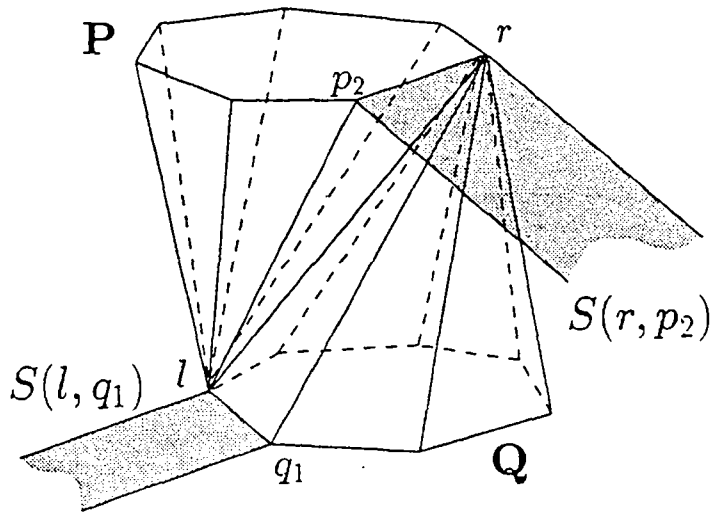


Figure A.2: The shadow of $\overline{rp_2}$ and $\overline{lq_1}$.

line segment $\overline{rp_2}$ so that the shadow $S(r, p_s)$ contains no other vertex of P. If $S(r, p_2)$ contains no other vertex, we assume p_s to be the vertex p_2. We replace the triangle (r,p_2,l) by the two triangles (r,p_s,l) and (p_s,p_2,l). We repeat the same procedure for $S(l, q_1)$, and add eventually a Steiner point q_s onto the edge $\overline{lq_1}$ (see Figure A.3).

Now we consider the tetrahedron formed by the vertices l, q_s, p_s, r. Since the shadows of $\overline{lq_s}$ and $\overline{rp_s}$ are empty, there is no edge of $\wp$ inside this tetrahedron. The edge $\overline{p_s q_s}$ lies completely outside of $\wp$. Therefore, we can remove the faces (r, p_s, l) and (r, q_s, l) from $\wp$ and replace them by (r, q_s, p_s) and (q_s, p_s, l) (see Figure A.4). In other words: we close the fold between the faces (r, p_s, l) and (r, q_s, l) with the tetrahedron (p_s, r, q_s, l)

Because our definition of a simple closed polyhedron requires triangular faces (a 2-dimensional simplicial complex), we need to close $\wp$ by two planar triangulations of the inside of P and Q, called $T(P)$ and $T(Q)$. The complete polyhedron $\wp$ consists now of

$$\begin{aligned}&(\text{cone}(P, l) \setminus \text{face}(r, p_s, l)) \cup (\text{cone}(Q, r) \setminus \text{face}(r, q_s, l)) \\ &\quad \cup \text{face}(r, q_s, p_s) \cup \text{face}(q_s, p_s, l) \cup T(P) \cup T(Q).\end{aligned}$$

It is obvious that the polyhedron satisfies the intersection condition and that it is connected. To verify the third condition, let us consider the link of r. The triangles of $\wp$ sharing r as a vertex are:

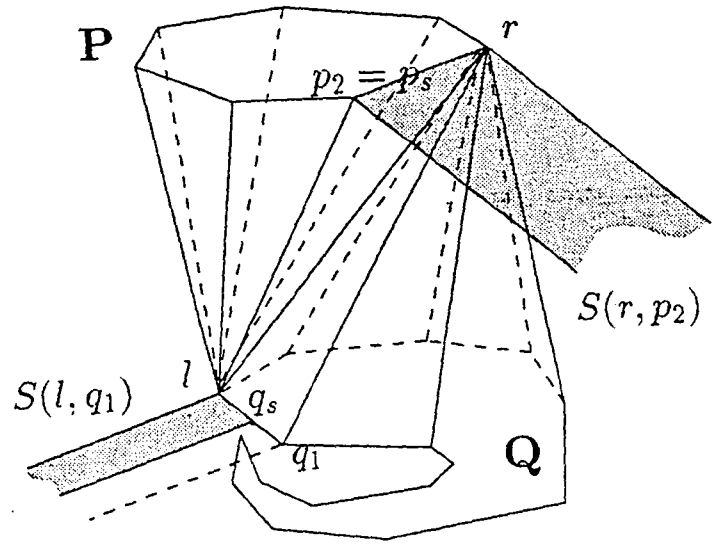


Figure A.3: The Steiner point q_s had been inserted, since $S(l, q_1)$ contained vertices of Q.

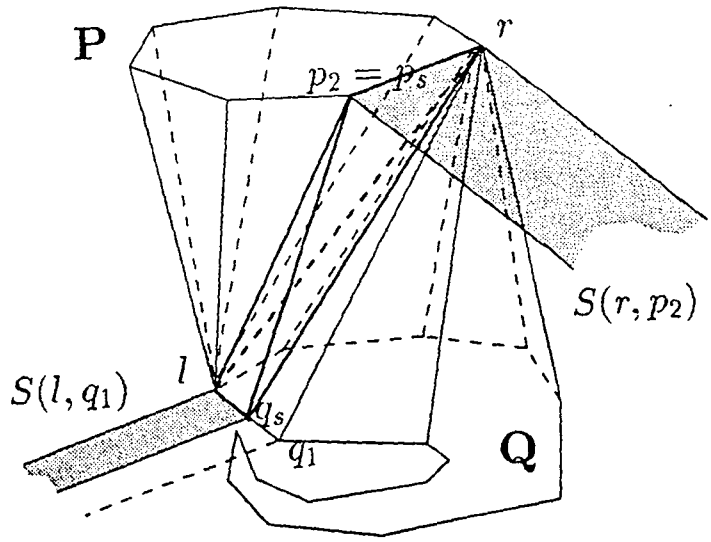


Figure A.4

a. all the faces of cone(Q, r) except face(r, q_s, l) which had been removed,

b. the face(l, r, p_1),

c. the portion of $T(P)$ having r as a vertex, say $T'(P)$ and

d. the face(r, p_s, q_s).

So the opposite edges of r are:

a. the polygon Q except edge $\overline{lq_s}$ which obviously is forming a simple polygonal arc starting at q_s and ending at l.

b. the edge $\overline{lp_1}$,

c. a simple polygonal arc starting in p_1 and ending in p_s which lies on the border of $T'(P)$. It depends on the particular triangulation method, but it must be simple, otherwise we would have overlapping triangles in $T'(P)$.

d. the edge $\overline{p_s q_s}$.

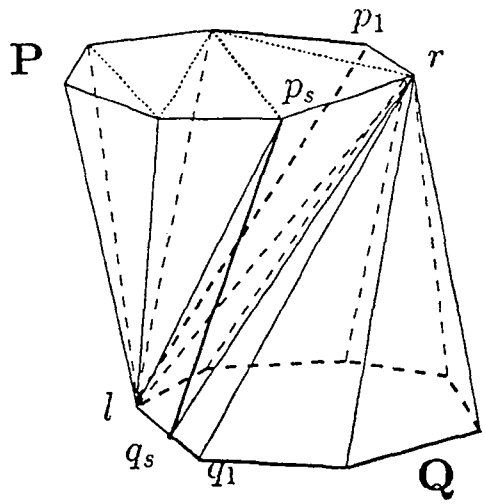


Figure A.5: The link of r.

Put together, these parts describe a non-planar simple closed polygon (see Figure A.5). It is easy to show that this condition is satisfied for all remaining vertices of $\wp$. Moreover, its intersection with any plane $z = z_t$, $(z_q \leq z_t \leq z_p)$ is a simple closed polygon:

$$\wp \cap \text{plane}(z = z_t) = F_p(\{P \setminus \overline{p_s r}\}, l) \cup F_q(\{Q \setminus \overline{l q_s}\}, r) \cup F_p(\{\overline{p_s r}\}, q_s) \cup F_q(\{\overline{l q_s}\}, p_s)$$

with

$$F_p(X, v) = \{\overline{v_1' v_2'} \mid \overline{v_1 v_2} \in X, v_i' = v_i + (v - v_i)\frac{z_t - z_p}{z_q - z_p}, i = 1, 2\},$$

$$F_q(X, v) = \{\overline{v_1' v_2'} \mid \overline{v_1 v_2} \in X, v_i' = v_i + (v - v_i)\frac{z_t - z_q}{z_p - z_q}, i = 1, 2\},$$

$F_p(\{P \setminus \overline{p_s r}\}, l)$ and $F_q(\{Q \setminus \overline{l q_s}\}, r)$ are two simple polygonal arcs with one point (belonging to edge $\overline{rl}$) in common. The two other end points are joined by the two edges $F_p(\{\overline{p_s r}\}, q_s) \in \text{face}(q_s, p_s, r)$ and $F_q(\{\overline{l q_s}\}, p_s) \in \text{face}(q_s, p_s, l)$, which in turn have a vertex in common, lying on the edge $\overline{p_s, q_s}$. Since this edge is not intersecting the cones, as we showed above, the overall intersection is a simple closed polygon. □

Appendix B

Results of the accuracy test

B.1 Pixel contours

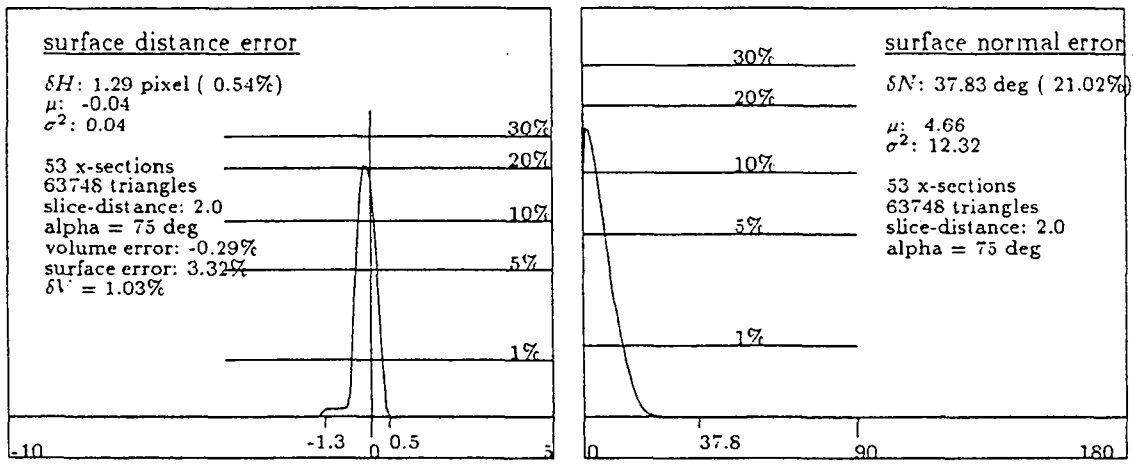


Figure B.1: Pixel contour at distance 2, 75deg

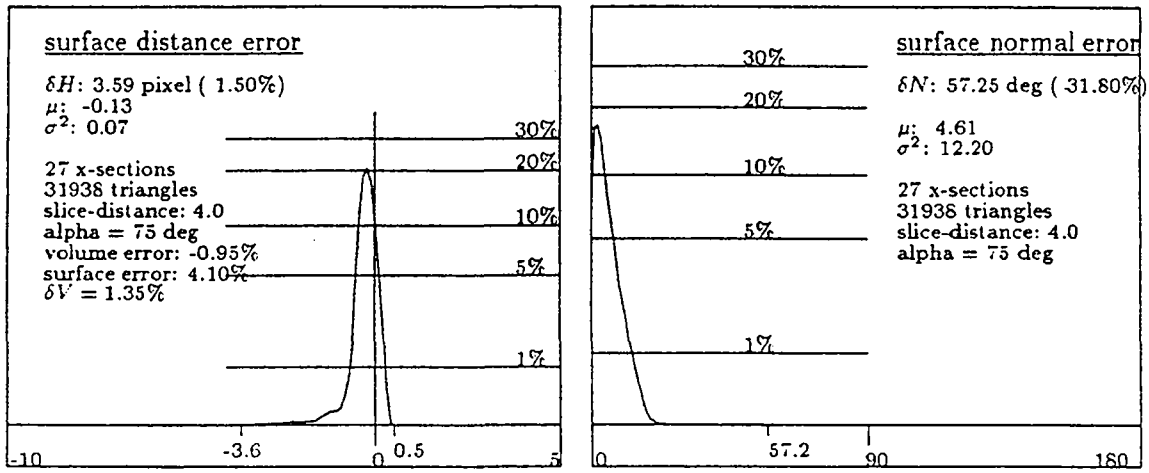


Figure B.2: Pixel contour at distance 4, 75deg

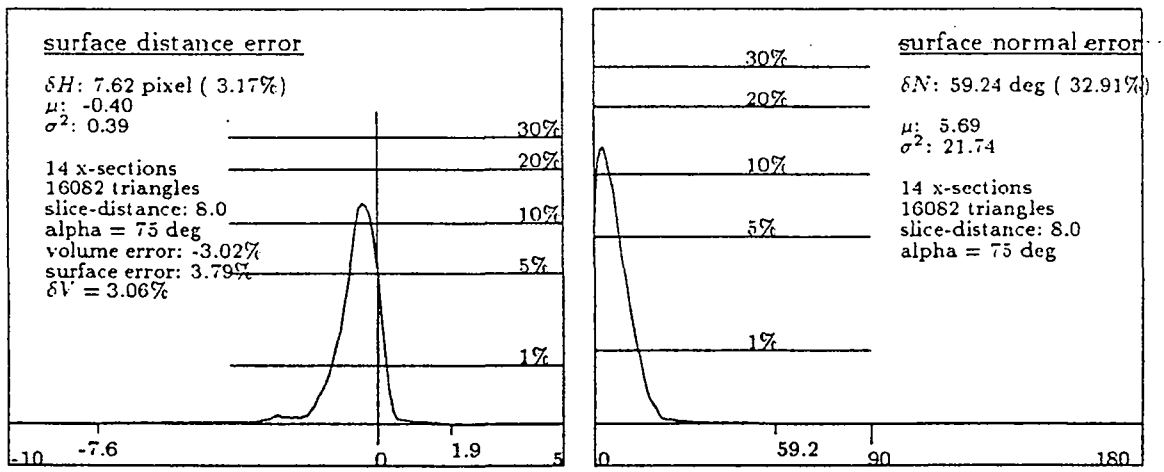


Figure B.3: Pixel contour at distance 8, 75deg

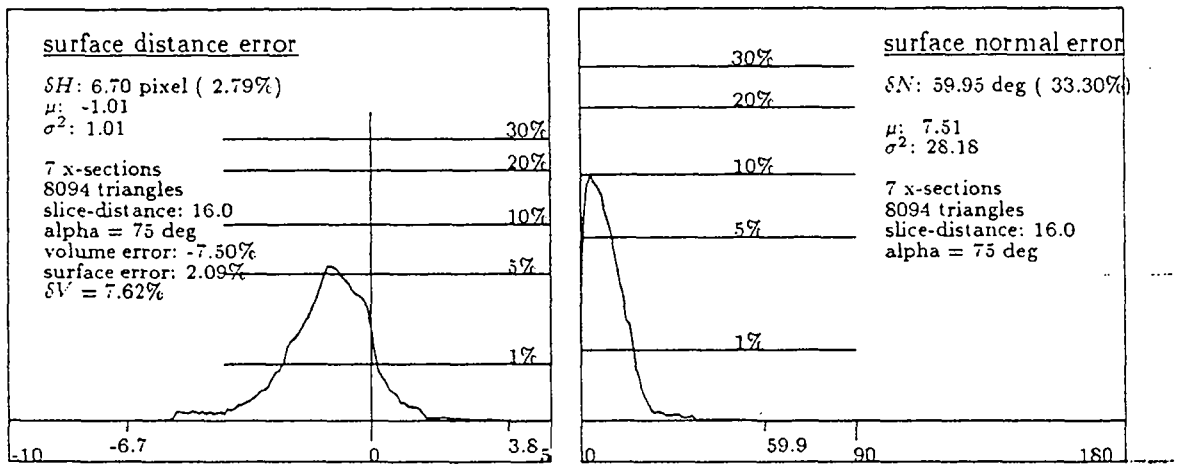


Figure B.4: Pixel contour at distance 16, 75deg

B.2 Vector contours

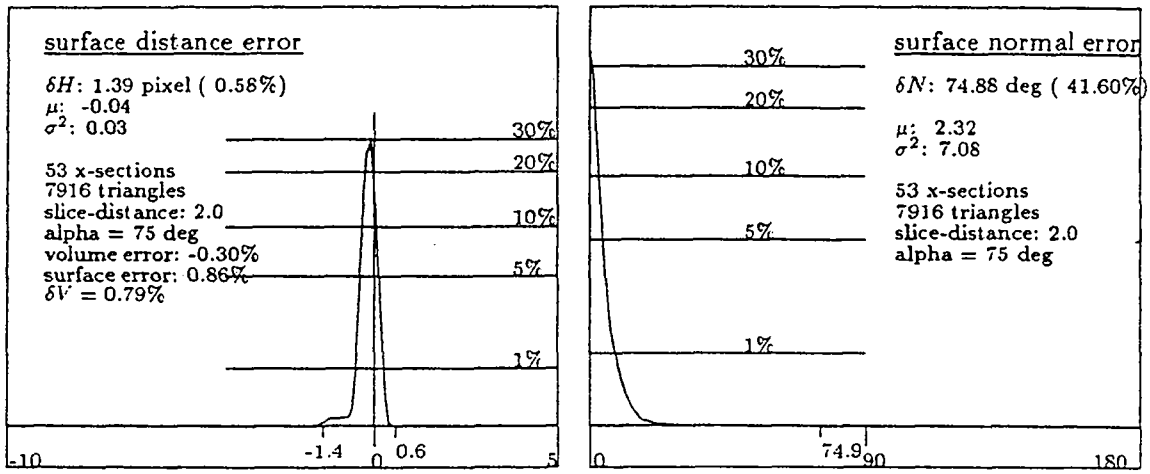


Figure B.5: Vector contour at distance 2, 75deg

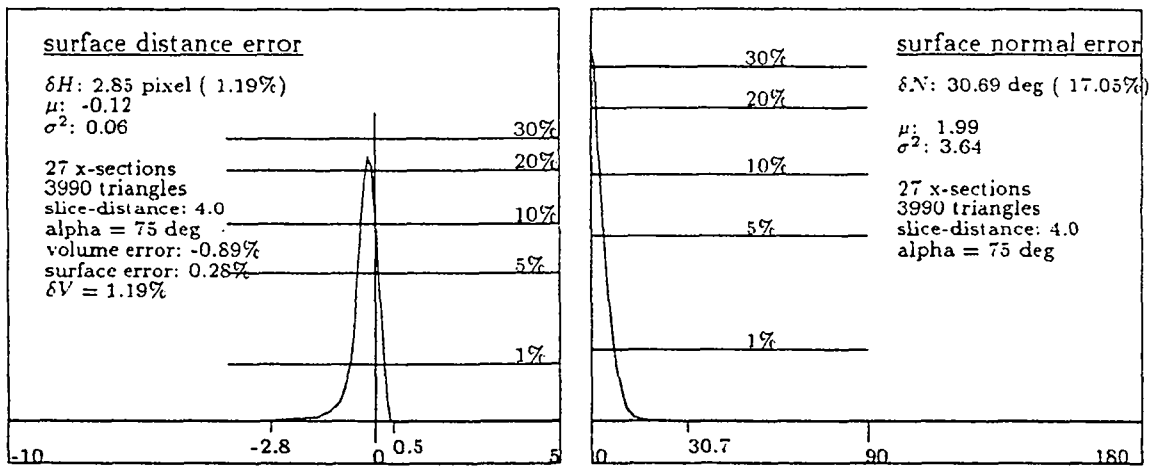


Figure B.6: Vector contour at distance 4, 75deg

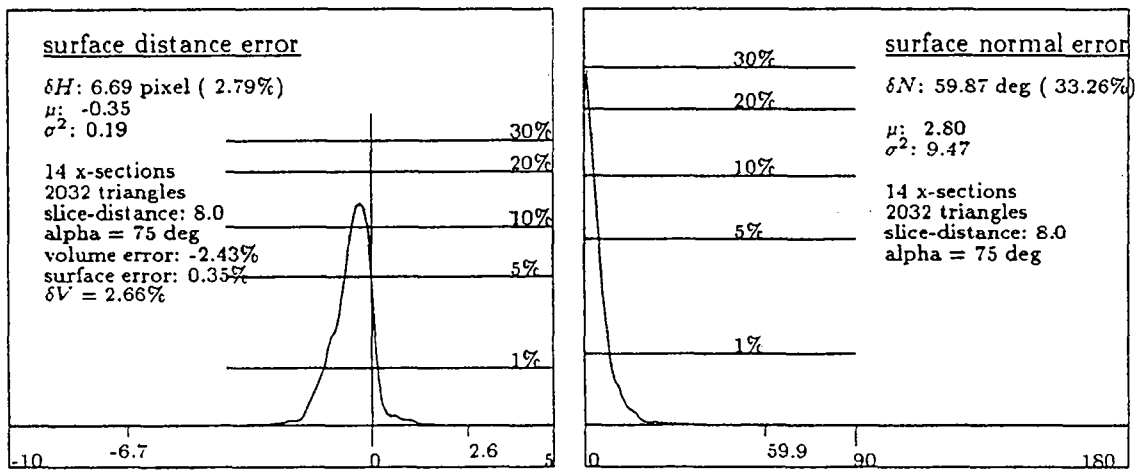


Figure B.7: Vector contour at distance 8, 75deg

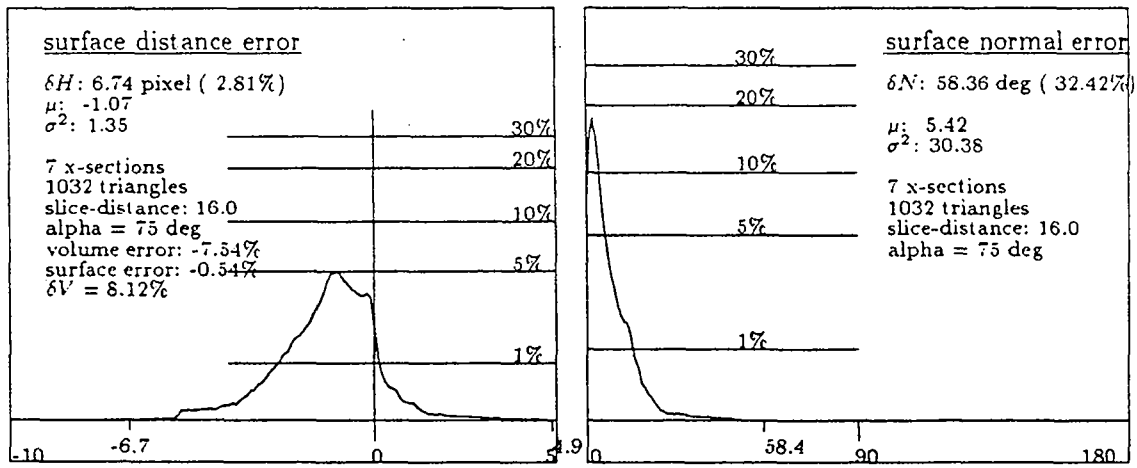


Figure B.8: Vector contour at distance 16, 75deg

B.3 Hand drawn contours

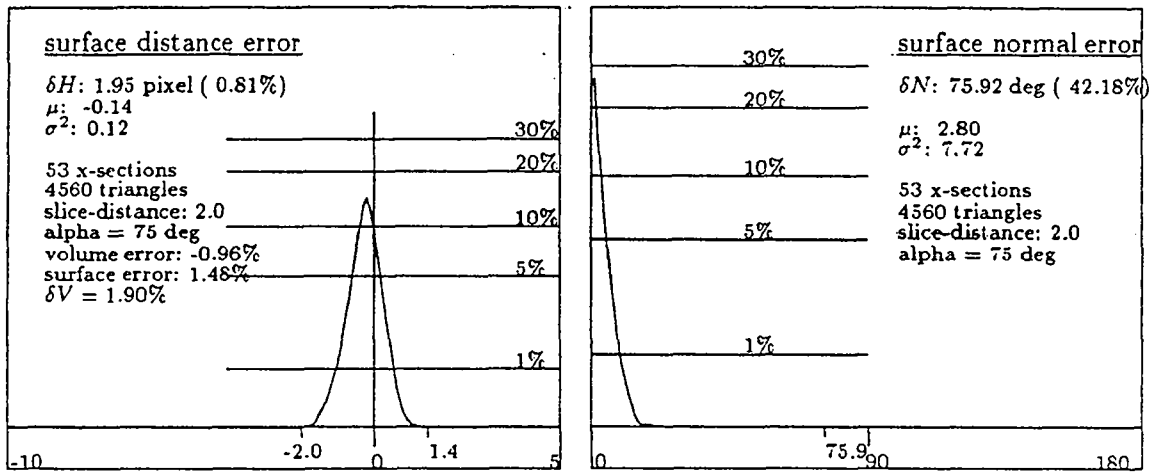


Figure B.9: Manually drawn contours at distance 2, 75deg

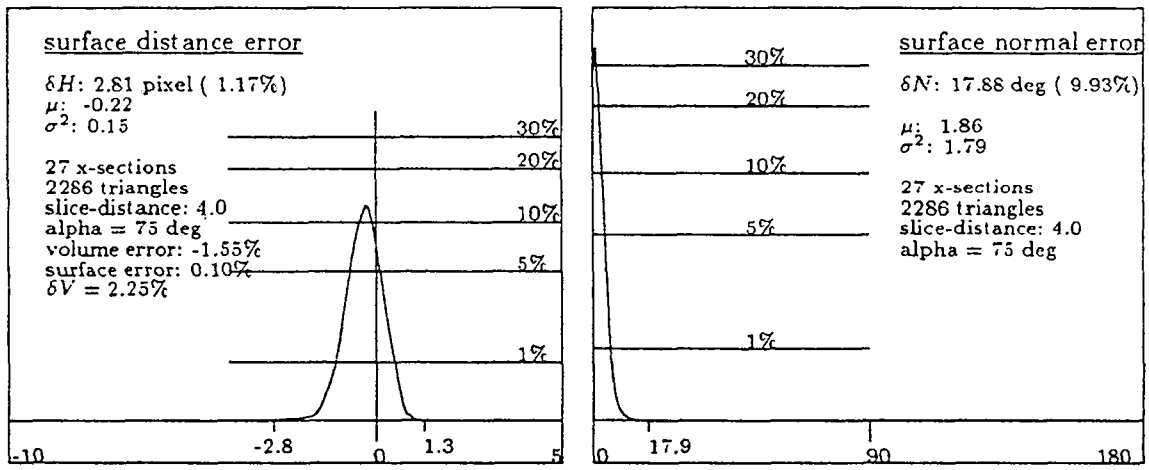


Figure B.10: Manually drawn contours at distance 4, 75deg

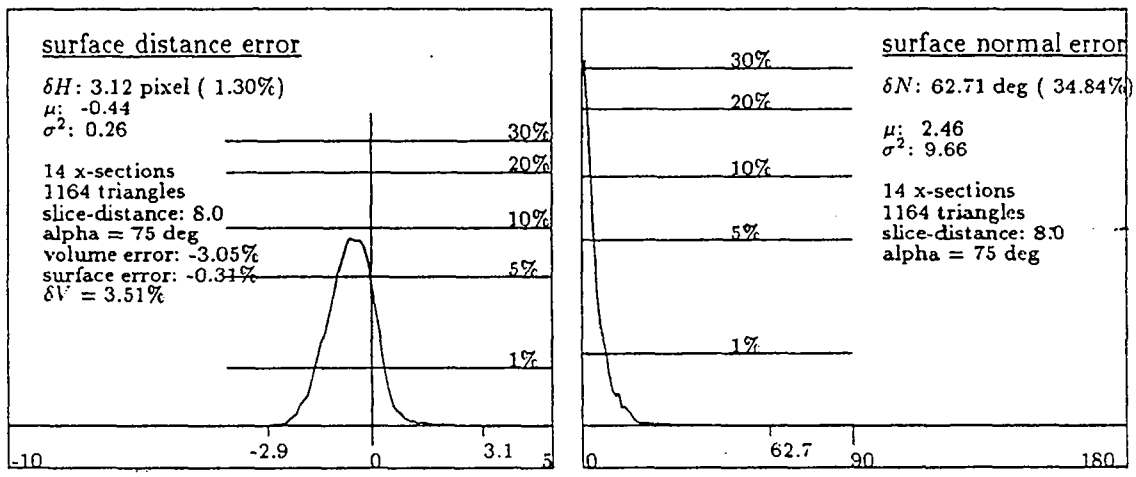


Figure B.11: Manually drawn contours at distance 8, 75deg

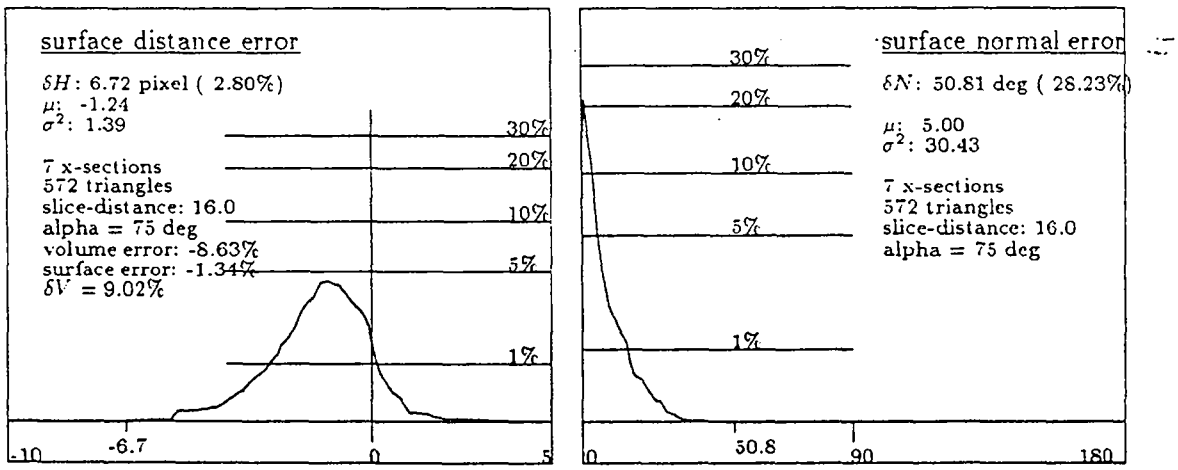


Figure B.12: Manually drawn contours at distance 16, 75deg

B.4 BBG interpolation

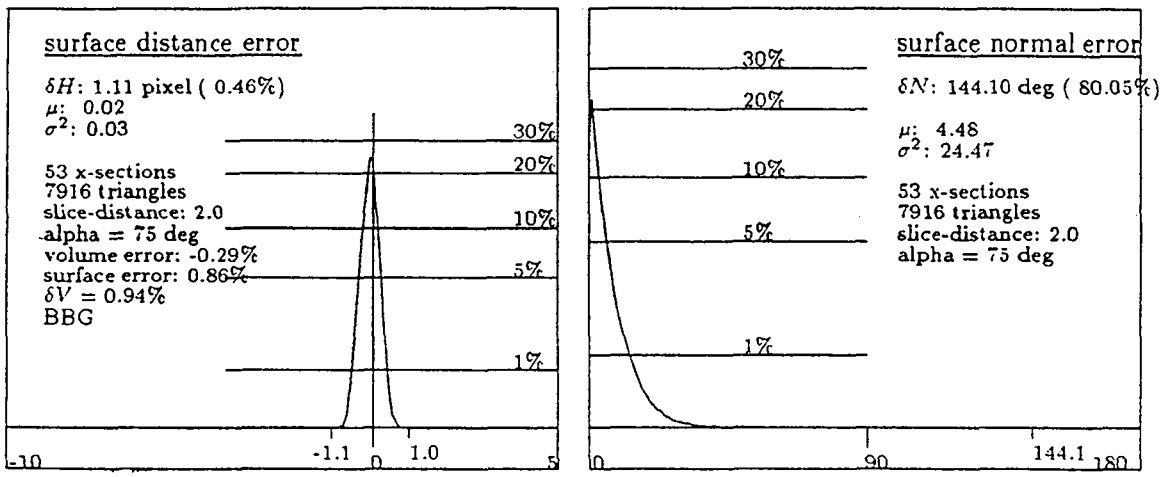


Figure B.13: Vector contour BBG distance 2, 75deg

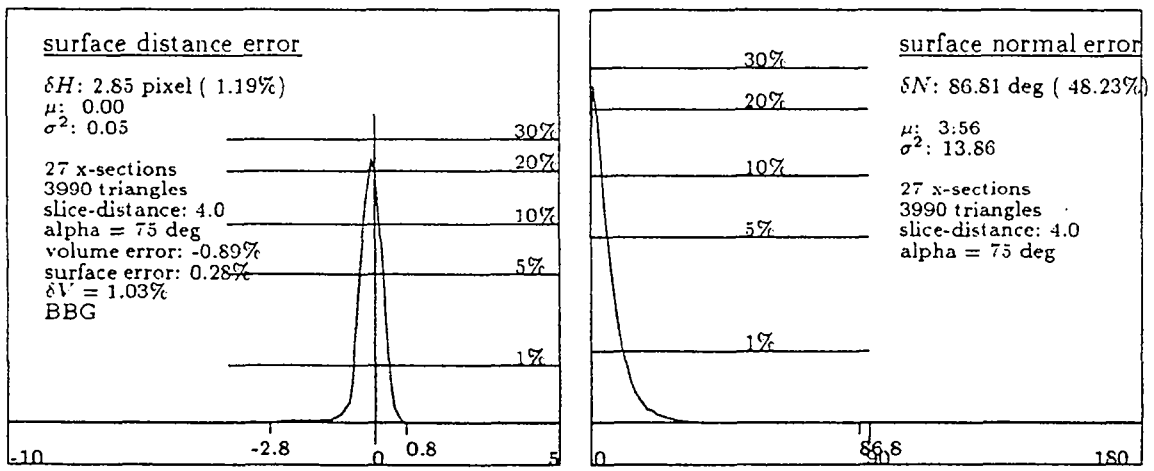


Figure B.14: Vector contour BBG distance 4, 75deg

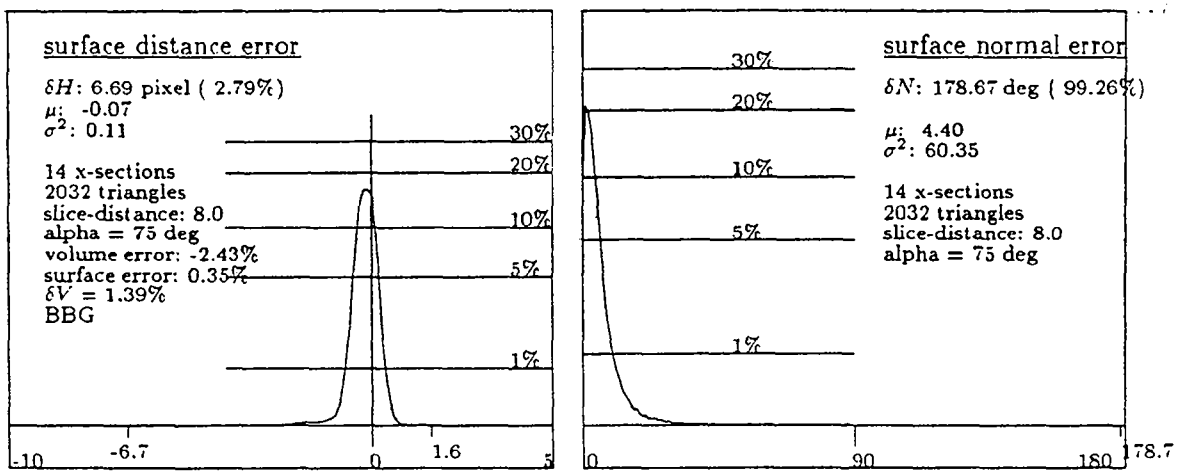


Figure B.15: Vector contour BBG distance 8, 75deg

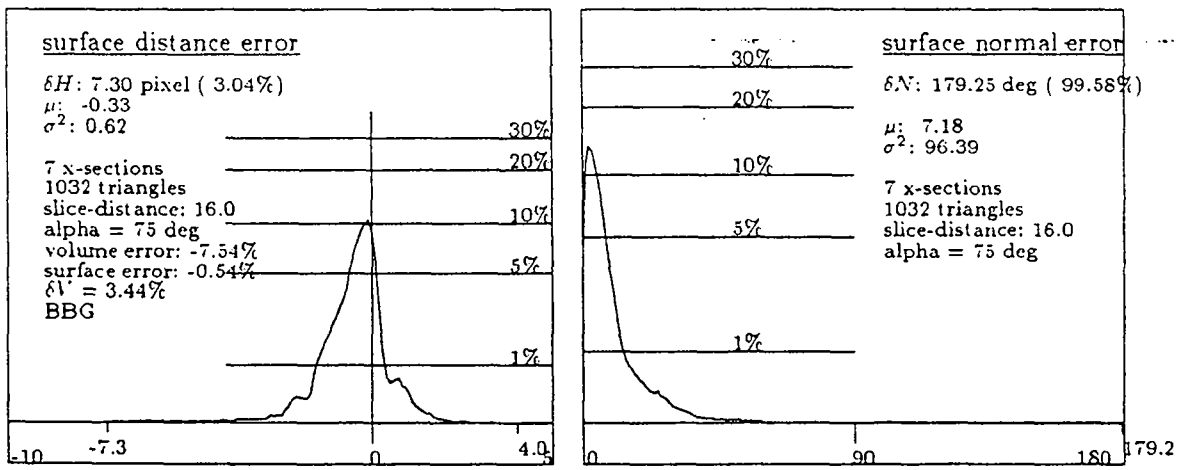


Figure B.16: Vector contour BBG distance 16, 75deg

B.5 Marching cube

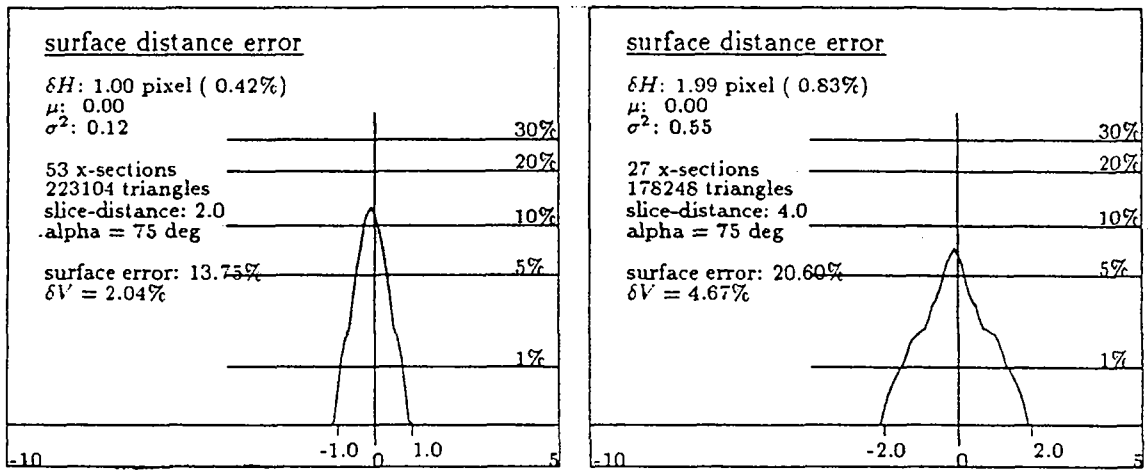


Figure B.17: Marching cube, distance 2 and 4, 75deg

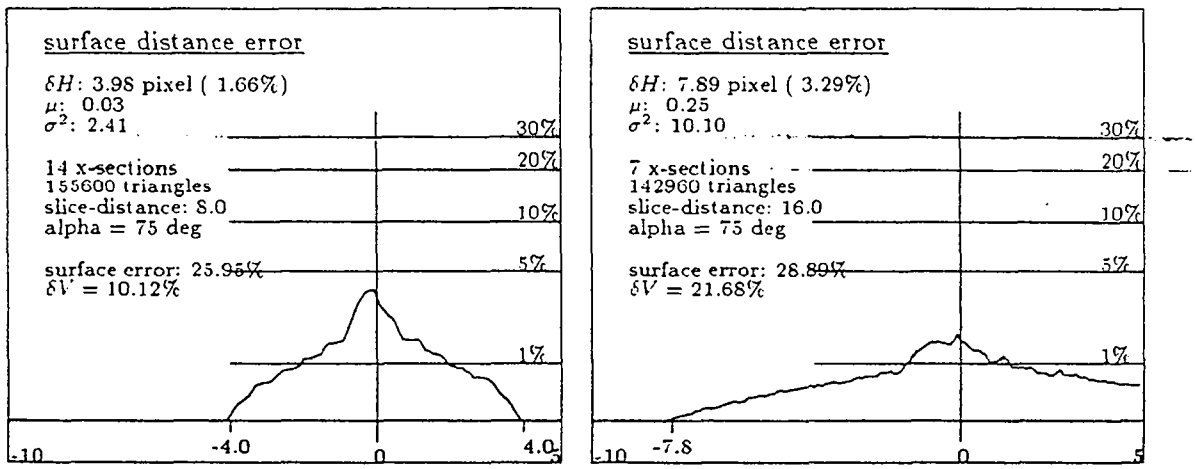


Figure B.18: Marching cube, distance 8 and 16,75deg

Appendix C

Glossary

biparietal diameter The largest diameter of the fetal head between the left hemisphere of the skull and the right hemisphere.

breech presentation the fetus lies with the buttocks next to the birth canal.

cerebrospinal fluid a liquid that is comparable to serum and is secreted from the blood into the lateral ventricles of the brain.

coccyx the end of the spinal column beyond the sacrum

MRI magnetic resonance imaging

occiput the back part of the head or skull

occiput presentation the fetus lies with the occiput closest to the birth canal (95% of cases).

presentation the position in which the fetus lies in the uterus with respect to the birth canal.

sagittal suture The suture between the parietal bones.

vertex presentation same as occiput presentation

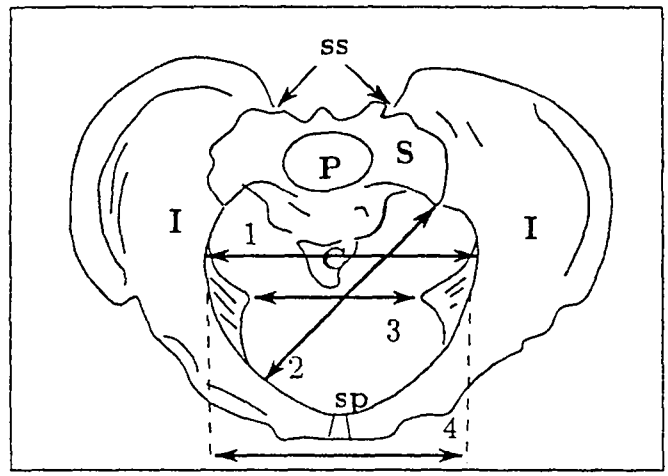


Figure C.1: The parts of the pelvis: sacrum (S), coccyx (C) and the two innominate bones (I) composed of ilium, ischium and pubis. The joints are the sacroilliac synchondroses (ss) and the symphysis pubis (sp). The promontory is marked with (P). The diameters are: transverse (inlet) (1), oblique (2), interspinous (3) and transverse median (4).

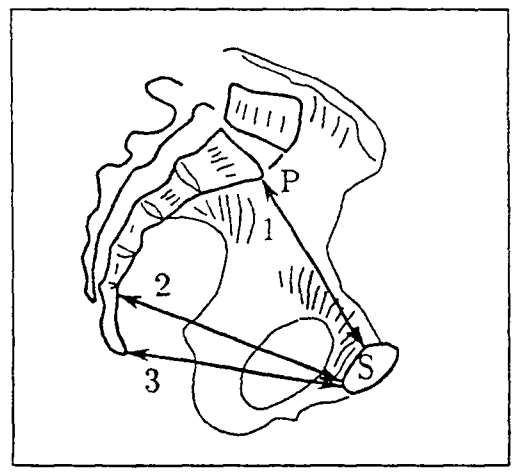


Figure C.2: A sagittal view of the pelvis with the anteroposterior diameters of inlet (obstetric conjugate) (1), midpelvis (2) and outlet (3).

List of Figures

Bibliography

Reconstruction

[1] F. Aurenhammer. Voronoi diagrams: a survey of a fundamental geometric data structure. *ACM Comput. Surv.*, 23:345–405, 1991.

[2] C. Barillot, B. Gilbaud, L. M. Luo, and J.M. Scarabin. 3-d representation of anatomic structures from CT examination. In *Biostereometrics*, pages 307–314, 1985.

[3] H. Blum. A transformation for extracting new descriptors of shape. In W. Walthen-Dunn, editor, *Models for the perception of speech and visual form*. MIT Press, Cambridge, MASS, 1967.

[4] J-D. Boissonnat. Shape reconstruction from planar cross-sections. *Computer Vision, Graphics, and Image Processing*, 44:1–29, 1988.

[5] J.-D. Boissonnat, A. Cérézo, O. Devillers, and M. Teillaud. Output-sensitive construction of the 3-d Delaunay triangulation of constrained sets of points. In *Proc. 3rd Canad. Conf. Comput. Geom.*, pages 110–113, 1991.

[6] J.-D. Boissonnat, A. Cérézo, O. Devillers, and M. Teillaud. Output-sensitive construction of the 3-d Delaunay triangulation of constrained sets of points. Research Report 1415, INRIA Sophia-Antipolis, Valbonne, France, April 1991.

[7] J-D. Boissonnat and B. Geiger. Three-dimensional reconstruction of complex shapes based on the Delaunay triangulation. Report 1697, INRIA Sophia-Antipolis, Valbonne, France, 1992.

[8] J-D. Boissonnat and B. Geiger. Three dimensional reconstruction of complex shapes based on the Delaunay triangulation. In R. S. Acharya and D. B. Goldgof, editors, *Biomedical Image Processing and Biomedical Visualization*, pages 964–975, San Jose CA, February 1993. SPIE. vol. 1905, part 2.

[9] A. Bowyer. Computing Dirichlet tesselations. *The Computer Journal*, 24(2), 1981.

[10] H. N. Christiansen and T. W. Sederberg. Conversion of complex contour line definitions into polygonal element mosaics. *Computer Graphics*, 8:658–660, August 1978.

[11] H. Edelsbrunner and T. S. Tan. An upper bound for conforming Delaunay triangulations. In *Proc. 8th Annu. ACM Sympos. Comput. Geom.*, pages 53–62, 1992.

[12] S. J. Fortune. Fast algorithms for polygon containment. In *LNCS 194 (12th Colloquium on Automata, Languages and Programming)*, pages 189–198. Springer-Verlag, 1985.

[13] H. Fuchs, Z. Kedem, and S.P. Uselton. Optimal surface reconstruction from planar contours. *Communications of the ACM*, 20:693–702, 1977.

[14] P.J.Giblin. *Graphs, Surfaces and Homology.* Chapman and Hall, London, 1977.

[15] P. J. Green and R. R. Sibson. Computing Dirichlet tessellations in the plane. *Comput. J.*, 21:168–173, 1978.

[16] F. Hermeline. Triangulation automatique d'un polyèdre en dimension n. *RAIRO Modél. Math. Anal. Numer.*, 16(3):211–242, 1982.

[17] K.H. Höhne and R. Bernstein. Shading 3d-images from CT using gray-level gradients. *IEEE Trans. Medical Imaging*, pages 45–47, 1986.

[18] E. Keppel. Approximating complex surfaces by triangulation of contour lines. *IBM Journal of res. and development*, 19:2–11, 1975.

[19] D.G. Kirkpatrick. Efficient computation of continuous skeletons. In *IEEE Symposium on Foundations of Computer Science*, volume 20, pages 18–27, 1979.

[20] A. Klingert and W. Straub. A robust and efficient triangulation algorithm for contours on parallel planes. Interner Bericht 11, Institut für Betriebs- und Dialogsysteme, Universität Karlsruhe, 1992.

[21] O. Kübler, F. Klein, R. Ogniewicz, and U. Kienholz. Isolation and identification of abutting and overlapping objects in binary images. In *5th internat. Conference on Image Analysis and Processing*, Positano, Italy, September 1989. World Scientific.

[22] M. Levoy. Display of surfaces from volume data. *IEEE Computer Graphics and Applications*, pages 29–37, 1988.

[23] W. Lorensen and H. Cline. Marching cubes: a high resolution 3d surface construction algorithm. *Computer Graphics*, 21:163–169, 1987.

[24] C. Gitlin, J. O'Rourke, and V. Subramanian. On reconstructing polyhedra from parallel slices. Technical Report 25, Dept. Comput. Sci., Smith College, Northampton, MA, March 1993.

[25] F. P. Preparata and M. I. Shamos. *Computational Geometry: an Introduction.* Springer-Verlag, New York, NY, 1985.

[26] Michel Schmitt. Some examples of algorithm analysis in computational geometry by means of mathematical morphological techniques. In Jean-Daniel Boissonnat and Jean-Paul Laumond, editors, *Geometry and Robotics*, LNCS 391. Springer-Verlag, May 1989.

[27] M. Shantz. Surface definition for branching contour defined objects. *Computer Graphics*, 15:242–270, July 1981.

[28] David Meyers, Shelley Skinner, and Kenneth Sloan. Surfaces from contours. *ACM Trans. Graph.*, 11(3):228–258, July 1992.

[29] K. D. Toennies. Surface triangulation by linear interpolation in intersection planes. In *Science and Engineering of Medical Imaging*, pages 98–105, 1989.

[30] C.K. Yap. An $O(n \log n)$ algorithm for the Voronoi diagram of a set of simple curve segments. *Discrete and Computational Geometry*, 2:365–393, 1987.

Verification of Accuracy

[31] S. Abramowski and H. Müller. *Geometrisches Modellieren.* BI Wiss.-Verl., 1991.

[32] H. Alt and M. Godau. Measuring the resemblance of polygonal curves. In *Proc. 8th Annu. ACM Sympos. Comput. Geom.*, pages 102–109, 1992.

[33] A. Klingert and B. Geiger. Von Konturen zu Oberflächenmodellen – Bewertung von Visualisierungsalgorithmen. In *Workshop Visualisierung in der Medizin*, GI, 1993.

Simulation of Delivery

[34] F. Avnaim and J-D. Boissonnat. Polygon placement under translation and rotation. *Informatique théorique et Applications/Theoretical Informatics and Applications*, 23(1):5–28, 1989.

[35] B. Geiger. Three dimensional simulation of delivery for cephalopelvic disproportion. In *First international workshop on mechatronics in medicine and surgery*, pages 146–152, Costa del Sol, October 1992.

[36] M. L. Gimovsky, K. Willard, M. Neglio, T. Howard, and S. Zerne. X-ray pelvimetry in a breech protocol: A comparison of digital radiography and conventional methods. *American Journal of Obstetrics and Gynecology*, 153:887–888, 1985.

[37] I. R. Johnson, E. M. Symonds, D. M. Kean, B. S. Worthington, F. Broughton Pipkin, R. C. Hawkes, and M. Gyngell. Imaging the pregnant human uterus with nuclear magnetic resonance. *American Journal of Obstetrics and Gynecology*, 148:1136–1139, 1984.

[38] J. N. Kopelman, P. Duff, R. T. Karl, A. H. Schipul, and J. A. Read. Computed tomographic pelvimetry in the evaluation of breech presentation. *Obstetrics & Gynecology*, 68(4):455–458, October 1986.

[39] T. A. Mahmood and J. M. Grant. The role of radiological pelvimetry in the management of patients who have had a previous caesarean section. *Journal of Obstetrics and Gynaecology*, 8(1):24–28, 1987.

[40] J. Mandry, H. Grandjean, J. M. Reme, J. Pastor, C. Levade, and G. Pontonnier. Assessment of the predictive value of X-ray pelvimetry and biparietal diameter in cephalopelvic disproportion. *Europ. J. Obstet. Gynec. reprod. Biol.*, 15:173–179, 1983.

[41] M.T. Mason. Compliant motion. In Brady & al, editor, *Robot , Motion-Planning and Control*. The MIT Press, Cambridge, 1982.

[42] S. M. McCarthy, R. A. Filly, D. D. Stark, H. Hricak, M. N. Brant-Zawadzki, P. W. Callen, and C. B. Higgins. Obstetric magnetic resonance imaging: Fetal anatomy. *Radiology*, 154:427–432, 1985.

[43] R. Merger, J. Levy, and J. Melchior. *Précis d'obstétrique*. Masson, 1979.

[44] J-P. Merlet. Use of c-surface based force-feedback algorithm for complex assembly tasks. In *ISER*, Toulouse, July 1991.

[45] M. T. Parson and W. N. Spellacy. Prospective randomized study of X-ray pelvimetry in the primigravida. *Obstetrics & Gynecology*, 66(1):76–79, July 1985.

[46] M. C. Powell, B. S. Worthington, J. M. Buckley, and E. M. Symonds. Magnetic resonance imaging (mri) in obstetrics. I. maternal anatomy. *British Journal of Obstetrics and Gynaecology*, 95:31–37, January 1988.

[47] J. A. Pritchard, P. C. MacDonald, and N. F. Gant. *Williams Obstetrics.* Appleton-Century-Crofts, 1985.

[48] J.T. Schwartz and M. Sharir. Algorithmic motion planning. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, pages 391–430. Elsevier Science Publishers B.V., 1990.

[49] Eliot L. Silbar. Factors related to the increasing cesarean section rates for cephalopelvic disproportion. *American Journal of Obstetrics and Gynecology*, pages 1095–1098, May 1986.

[50] F. W. Smith, C. Kent, D. R. Abramovich, and H. W. Sutherland. Nuclear magnetic resonance imaging—a new look at the fetus. *British Journal of Obstetrics and Gynaecology*, 92:1024–1033, October 1985.

[51] J-C. Sournia. *Histoire de la médecine et des médecins.* Larousse, Paris, 1991.

[52] D. D. Stark, S. M. McCarthy, R. A. Filly, J. T. Parer, H. Hricak, and P. W. Callen. Pelvimetry by magnetic resonance imaging. *AJR*, 144:947–950, May 1985.

[53] D. Vanel and M. T. McNamara, editors. *MRI of the body*, pages 363–368. Springer-Verlag, 1989.

[54] J. C. Weinreb, T. W. Lowe, R. Santos-Ramos, F. G. Cunningham, and R. Parkey. Magnetic resonance imaging in obstetric diagnosis. *Radiology*, 154:157–161, 1985.

Unité de Recherche INRIA Sophia Antipolis
2004, route des Lucioles - B.P. 93 - 06902 SOPHIA ANTIPOLIS Cedex (France)

Unité de Recherche INRIA Lorraine Technopôle de Nancy-Brabois - Campus Scientifique
615, rue du Jardin Botanique - B.P. 101 - 54602 VILLERS LES NANCY Cedex (France)
Unité de Recherche INRIA Rennes IRISA, Campus Universitaire de Beaulieu 35042 RENNES Cedex (France)
Unité de Recherche INRIA Rhône-Alpes 46, avenue Félix Viallet - 38031 GRENOBLE Cedex (France)
Unité de Recherche INRIA Rocquencourt Domaine de Voluceau - Rocquencourt - B.P. 105 - 78153 LE CHESNAY Cedex (France)

EDITEUR
INRIA - Domaine de Voluceau - Rocquencourt - B.P. 105 - 78153 LE CHESNAY Cedex (France)

ISSN 0249 - 6399